comput. complex. **27** (2018), 1 – 29

1016-3328/18/010001-29
published online November 30, 2016
DOI 10.1007/s00037-018-0148-5

computational complexity

ON THE HARDNESS OF THE NONCOMMUTATIVE DETERMINANT

V. Arvind and Srikanth Srinivasan

Abstract. In this paper, we study the computational complexity of computing the *noncommutative* determinant. We first consider the arithmetic circuit complexity of computing the noncommutative determinant polynomial. Then, more generally, we also examine the complexity of computing the determinant (as a function) over noncommutative domains. Our hardness results are summarized below:

- We show that if the noncommutative determinant polynomial has small noncommutative arithmetic circuits then so does the noncommutative permanent. Consequently, the commutative permanent polynomial has small commutative arithmetic circuits.
- For any field $\mathbb{F}$ we show that computing the $n \times n$ permanent over $\mathbb{F}$ is polynomial-time reducible to computing the $2n \times 2n$ (noncommutative) determinant whose entries are $O(n^2) \times O(n^2)$ matrices over the field $\mathbb{F}$.
- We also derive as a consequence that computing the $n \times n$ permanent over nonnegative rationals is polynomial-time reducible to computing the noncommutative determinant over Clifford algebras of $n^{O(1)}$ dimension.

Our techniques are elementary and use primarily the notion of the *Hadamard Product* of noncommutative polynomials.

Keywords. Determinant; Noncommutative arithmetic circuits; Permanent; Permanent estimators.

Subject classification. 68Q17

1. Introduction

In a breakthrough paper (Nisan 1991), Nisan systematically studied the problem of proving lower bounds for noncommutative computation. The focus of his study was noncommutative arithmetic circuits, noncommutative arithmetic formulas and noncommutative algebraic branching programs. In his central result based on a rank argument, Nisan shows that the noncommutative permanent or determinant polynomials in the ring $\mathbb{F}\langle x_{11}, \ldots, x_{nn}\rangle$ require exponential size noncommutative algebraic branching programs.

Nisan's results are over the *free* noncommutative ring $\mathbb{F}\langle X\rangle$. Chien & Sinclair (2007) explore the same question over other noncommutative algebras. They refine Nisan's rank argument to show exponential size lower bounds for formulas computing the permanent or determinant over specific noncommutative algebras, like the algebra of 2×2 matrices over $\mathbb{F}$, the quaternion algebra, and a host of other examples.

However, the question of whether there is a small noncommutative *circuit* for the determinant or permanent remains unanswered. (Indeed, no explicit lower bounds are known for the general noncommutative circuit model.) Since the existence of small noncommutative arithmetic circuits for the permanent would imply the existence of small *commutative* arithmetic circuits for the permanent, we have a good reason to believe that the permanent does not have small noncommutative arithmetic circuits. However, as far as we know, before this work, no such argument has been given for the case of the noncommutative *determinant*. Indeed, since Nisan (1991) has also shown an exponential separation between the power of noncommutative formulas and circuits, it may very well be that the noncommutative determinant has polynomial-sized arithmetic circuits.

Another motivation for studying the computational difficulty of computing the noncommutative determinant (as a function) is an approach to designing randomized approximation algorithms for the 0-1 permanent by designing good unbiased estimators based on the determinant. This approach has a long history starting with Godsil & Gutman (1981) and Karmarkar *et al.* (1993). Of specific interest are the works of Barvinok (2000); Chien *et al.*

(2003) and more recently that of Moore & Russell (2012). Barvinok (2000) defines a variant of the noncommutative determinant called the *symmetrized determinant* and shows that given inputs from a *constant-dimensional* matrix algebra, the symmetrized determinant over these inputs can be evaluated in polynomial time. He uses these to define a series of algorithms that he conjectures might yield progressively better randomized approximation algorithms for the (commutative) permanent. Chien, Rasmussen, and Sinclair (Chien *et al.* 2003) show that efficient algorithms to compute the determinant over Clifford algebras of polynomial dimension would yield efficient approximation algorithms for the permanent. Moore & Russell (2012) provide evidence that Barvinok's approach might not work, but their results also imply that computing the symmetrized or standard noncommutative determinant over polynomial-dimensional matrix algebras would give a good estimator for the permanent.

Our results.

1. We provide evidence that the noncommutative determinant is hard. We show that if the noncommutative determinant[1] can be computed by a small noncommutative arithmetic circuit, then so can the noncommutative permanent and therefore, the commutative permanent has small commutative arithmetic circuits. This is in marked contrast to the commutative case, where the determinant is known to be computable by polynomial-sized circuits, but the permanent is not known (or expected) to have subexponential-sized arithmetic circuits.

2. We show that computing the noncommutative determinant over matrix algebras of polynomial dimension is as hard as computing the commutative permanent. We also derive as a consequence that computing the $n \times n$ permanent over nonnegative rationals is polynomial-time reducible to computing

[1] We have not yet formally defined the noncommutative determinant and there are, in fact, many ways of doing it. See Section 2.

the noncommutative determinant over Clifford algebras of poly(n) dimension.

This points to the intractability of carrying over Barvinok's approach for large dimension, and also to the possibility that the approach of Chien, Rasmussen, and Sinclair might be computationally infeasible.

We stress that our result here is potentially more useful than a noncommutative circuit lower bound for the determinant, from an algorithmic point of view. This is because an arithmetic circuit lower bound result would not rule out the possibility of a polynomial-time algorithm for the noncommutative determinant over even polynomial dimension matrix algebras. For example, Barvinok's algorithm (Barvinok 2000) computes the symmetrized determinant over constant-dimensional matrix algebras, whereas any algebraic branching program that computes the symmetrized determinant over constant-dimensional matrix algebras must be of exponential size (Chien & Sinclair 2007).

Independent work of Hrubeš, Wigderson, and Yehudayoff (Hrubeš *et al.* 2010). In an independent work, Hrubeš *et al.* (2010) consider arithmetic circuits over domains which may be noncommutative as well as *non-associative*. By adapting the proof method of Valiant, they show that the permanent, when suitably defined, remains "complete" for the class VNP in this world. It follows from this that the permanent is VNP-complete in the noncommutative setting as well. This gives a converse of our main theorem statement i.e. that if the noncommutative permanent has small arithmetic circuits, then so does the noncommutative determinant.[2]

For the determinant, it is shown that they are hard for the class of polynomials computable by polynomial-sized *formulas*.

[2] There is a small caveat here, since the result of Hrubeš *et al.* (2010) is stated for one of the flavours of the noncommutative determinant and permanent, namely the *Cayley* variant, but our result holds for some others as well.

Subsequent work. Subsequent to a preliminary version of our results, there have been a few other results that proved hardness results for computing the noncommutative determinant. Chien *et al.* (2011) and Bläser (2015) showed that the problem of computing the determinant over many *constant-dimensional* algebras, such as constant-dimensional matrix algebras, is #P-hard if the field is $\mathbb{Q}$ and Mod_kP-hard if the field has constant characteristic $k > 0$; both these results were proved by suitably modifying #P-hardness proofs for the permanent. An alternate proof, by using ideas from the proof of Barrington's theorem (Barrington 1989), was recently found by Gentry (2014).

While the above proofs work in some settings where our proofs do not, the theorems we present still have merit for many reasons. For one, the proofs of the theorems above work for a particular definition of the noncommutative determinant (specifically the *Cayley* determinant) and not others: indeed, one of the variants we consider (the symmetrized determinant) can be computed in polynomial time over constant-dimensional matrix algebras. Secondly, it is unclear if these results also give the same consequences for noncommutative arithmetic circuits that we get here. Finally, the proofs we give below, especially in the arithmetic circuit setting, are, in our opinion, considerably simpler than the proofs of the above-mentioned theorems.

2. Preliminaries

For any set of variables X, let $\mathbb{F}\langle X\rangle$ denote the ring of *noncommuting* polynomials over X. Let $\mathcal{M}(X)$ denote the set of noncommutative monomials over X; given $d \in \mathbb{N}$, let $\mathcal{M}_d(X)$ denote the monomials over X of degree exactly d. For $f \in \mathbb{F}\langle X\rangle$ and $m \in \mathcal{M}(X)$, we will denote by $f[m]$ the coefficient of the monomial m in f.

For any ring R, we use $M_n(R)$ to denote the ring of $n \times n$ matrices with entries from R.

Fix $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, two disjoint sets of variables. Given $f \in \mathbb{F}\langle X\rangle$, matrices $A_i \in M_k(\mathbb{F}\langle Y\rangle)$ for $1 \leq i \leq m$, and $i_0, j_0 \in [k]$, we use $f(A_1, A_2, \ldots, A_m)(i_0, j_0)$ to denote the (i_0, j_0)th entry of the matrix $f(A_1, A_2, \ldots, A_m) \in M_k(\mathbb{F}\langle Y\rangle)$.

2.1. Noncommutative determinants and permanents.
Given $X = \{x_{ij} \mid 1 \leq i, j \leq n\}$ for $n \in \mathbb{N}$, we define the $n \times n$ noncommutative determinant and permanent polynomials over the set of variables X. By fixing the order of multiplication in each monomial of the *commutative* determinant/permanent polynomials in different ways, one can obtain many different reasonable ways of defining the $n \times n$ noncommutative determinant and permanent, and indeed many of these definitions have been studied (see (Aslaksen 2009), which surveys various flavours of the noncommutative determinant). The most straightforward definitions are those of the *Cayley determinant* and *Cayley permanent*—we will denote these by $\mathrm{Cdet}_n(X)$ and $\mathrm{Cperm}_n(X)$, respectively—which use the row order of multiplication. That is,

$$\mathrm{Cdet}_n(X) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma)\ x_{1,\sigma(1)} \cdot x_{2,\sigma(2)} \cdots x_{n,\sigma(n)},$$
$$\mathrm{Cperm}_n(X) = \sum_{\sigma \in S_n} x_{1,\sigma(1)} \cdot x_{2,\sigma(2)} \cdots x_{n,\sigma(n)}.$$

We also define the *Moore determinant* and *Moore permanent*—denoted $\mathrm{Mdet}_n(X)$ and $\mathrm{Mperm}_n(X)$, respectively—by ordering the variables in each monomial using the cyclic order of the corresponding permutation. Given $\sigma \in S_n$, we can write it uniquely as a product of r^σ disjoint cycles $(n^\sigma_{11} \cdots n^\sigma_{1l_1})(n^\sigma_{21} \cdots n^\sigma_{2l_2}) \cdots (n^\sigma_{r^\sigma 1} \cdots n^\sigma_{r^\sigma l_{r^\sigma}})$ such that for any $i \in [r^\sigma]$ and $j \in [l_i] \setminus \{1\}$, we have $n^\sigma_{i1} < n^\sigma_{ij}$ and $n^\sigma_{11} > n^\sigma_{21} > \cdots > n^\sigma_{r^\sigma 1}$. The Moore determinant and permanent are defined as

$$\mathrm{Mdet}_n(X) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma)\ x_{n^\sigma_{11},n^\sigma_{12}} \cdots x_{n^\sigma_{1l_{r^\sigma}},n^\sigma_{11}} \cdots x_{n^\sigma_{r^\sigma 1},n^\sigma_{r^\sigma 2}} \cdots x_{n^\sigma_{r^\sigma l_{r^\sigma}},n^\sigma_{r^\sigma 1}},$$
$$\mathrm{Mperm}_n(X) = \sum_{\sigma \in S_n} x_{n^\sigma_{11},n^\sigma_{12}} \cdots x_{n^\sigma_{1l_{r^\sigma}},n^\sigma_{11}} \cdots x_{n^\sigma_{r^\sigma 1},n^\sigma_{r^\sigma 2}} \cdots x_{n^\sigma_{r^\sigma l_{r^\sigma}},n^\sigma_{r^\sigma 1}}.$$

In the setting of a field $\mathbb{F}$ of characteristic 0, Alexander Barvinok, in Barvinok (2000), has studied another variant of the noncommutative determinant called the *symmetrized determinant*, which is denoted $\mathrm{sdet}_n(X)$. It is defined as follows:

$$\mathrm{sdet}_n(X) = \frac{1}{n!} \sum_{\sigma,\tau \in S_n} \mathrm{sgn}(\sigma)\mathrm{sgn}(\tau)\ x_{\tau(1),\sigma(1)} x_{\tau(2),\sigma(2)} \cdots x_{\tau(n),\sigma(n)}.$$

Barvinok (2000) shows that, for any fixed-dimensional associative algebra $\mathcal{A}$ over $\mathbb{F}$ of characteristic zero, there is a polynomial-time algorithm which, on input an $n \times n$ matrix A with entries from $\mathcal{A}$, computes $\mathrm{sdet}_n(A)$. It is not known whether such algorithms exist for the Cayley or Moore determinants.

2.2. Models for noncommutative arithmetic computation. A *noncommutative arithmetic circuit* C over a field $\mathbb{F}$ is defined as follows: C is a directed acyclic graph and every leaf of the graph is labelled with either an input variable from the set of variables X or an element from $\mathbb{F}$. Every internal node is labelled by either $(+)$ or $(\times)$—meaning that it is either an addition or multiplication gate, respectively—and has fanin two. Since we are working over noncommutative domains, we will assume that each multiplication gate has a designated left child and a designated right child. Each gate of the circuit computes a polynomial in $\mathbb{F}\langle X\rangle$ in the natural way: the polynomials computed at the leaves are the polynomials labelling the leaves; the polynomial computed at an internal node labelled by $+$ (resp. $\times$) is the sum (resp. product in left-to-right order) of the polynomials computed at its children. The polynomial computed by C is the polynomial computed at a designated output node of the circuit.

We also recall the definition of an algebraic branching program (ABP) computing a noncommutative polynomial in $\mathbb{F}\langle X\rangle$ (Nisan (1991); Raz & Shpilka (2005)). An ABP is a directed acyclic graph with one vertex of in-degree zero, which is called the *source*, and one vertex of out-degree zero, which is called the *sink*. The vertices of the graph are partitioned into levels numbered $0, 1, \ldots, d$. Edges may only go from level i to level $i+1$ for $i = 0, 1, \ldots, d-1$. The source is the only vertex at level 0, and the sink is the only vertex at level d. Each edge is labelled with a homogeneous linear form in the variables X. The size of the ABP is the number of vertices.

The ABP computes a degree d homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ as follows. Fix any path γ from source to sink with edges $e_1, e_2, \ldots, e_d$, where e_i is the edge from level $i-1$ to level i, and let ℓ_i denote the linear form labelling edge e_i. We denote by f_γ the homogeneous degree d polynomial $\ell_1 \cdot \ell_2 \cdots \ell_d$ (note that the order of multiplication is important). The polynomial f computed

by the ABP is simply

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

where $\mathcal{P}$ is the set of all paths from the source to the sink.

We will also consider a slight variant of the above definition where we allow multiple sources and sinks. A *multi-output ABP* P is defined exactly as above, except that we allow multiple sources at level 0 and multiple sinks at level d. For each source s and sink t, an ABP $P_{s,t}$ may be obtained from P by removing all sources other than s and sinks other than t. Let $S = \{s_1, s_2, \ldots, s_a\}$ and $T = \{t_1, t_2, \ldots, t_b\}$ denote the sets of sources and sinks, respectively in P. The ABP P will be thought of as computing the ab many polynomials computed by the ABPs P_{s_i,t_j}. More precisely, the output of the ABP P is an $a \times b$ matrix A with entries from $\mathbb{F}\langle X\rangle$ such that the (i,j)th entry of A, denoted $A(i,j)$, is the polynomial computed by the ABP P_{s_i,t_j}. It is easily seen that we can write A as $\sum_{m \in \mathcal{M}_d(X)} A_m m$, where $A_m \in \mathbb{F}^{a \times b}$; we will call A_m the *coefficient matrix* of the monomial m in the matrix A.

3. The Hadamard product

A key notion we require for all our reductions is the Hadamard product of polynomials that was introduced in Arvind *et al.* (2009).

Definition 3.1. *Given polynomials $f, g \in \mathbb{F}\langle X\rangle$, their* Hadamard product $h = f \circ g$ *is defined as follows: h is the unique polynomial in $\mathbb{F}\langle X\rangle$ such that for any monomial $m \in \mathcal{M}(X)$, the coefficient $h[m] = f[m] \cdot g[m]$.*

In Arvind *et al.* (2009, Theorem 5), it is shown how, given a noncommutative circuit for polynomial f and an ABP for polynomial g, we can efficiently compute a noncommutative circuit for their Hadamard product $f \circ g$. However, the construction in Arvind *et al.* (2009) modifies the noncommutative circuit for the polynomial f. Hence, it will not work if we are allowed only black-box access to f, which we require for certain applications in this paper.

Suppose we have an efficient *black-box* algorithm for evaluating the polynomial $f \in \mathbb{F}\langle X\rangle$, where the variables in X take values in

some matrix algebra (say, $n \times n$ matrices over a field $\mathbb{F}$). Furthermore, suppose we are given an ABP for the polynomial g. *Ideally,* we would like to obtain an efficient algorithm for computing their Hadamard product $f \circ g$ over the same matrix algebra.

However, what we can show is that we can put together the ABP and the black-box algorithm for f to obtain an efficient algorithm that computes $f \circ g$ over $\mathbb{F}$. This turns out to be sufficient to prove all our hardness results for the different noncommutative determinants.

THEOREM 3.2. *Fix $d \in \mathbb{N}$. Let $Z = \{z_1, z_2, \ldots, z_n\}$ be a set of noncommuting variables and $g \in \mathbb{F}\langle Z\rangle$ be a homogeneous polynomial of degree d such that g is computed by an ABP P of size S. Then, there exist matrices $A_1, A_2, \ldots, A_n \in M_S(\mathbb{F})$ such that for any homogeneous polynomial $f \in \mathbb{F}\langle Z\rangle$ of degree d, $f \circ g = f(A_1 z_1, A_2 z_2, \ldots, A_n z_n)(1, S)$. Moreover, given the ABP P, the matrices $A_1, A_2, \ldots, A_n$ can be computed in time polynomial in the size of the description of P.*

PROOF (Theorem 3.2). Let the vertices of P be named $1, 2, \ldots, S$ where 1 is the source of the ABP and S is the sink. Define the matrices $A_1, A_2, \ldots, A_n \in M_S(\mathbb{F})$ as follows: $A_i(k, l)$ is the coefficient of the variable z_i in the linear form labelling the edge that goes from vertex k to vertex l; if there is no such edge, the entry $A_i(k, l) = 0$. For any monomial $m = z_{i_1} z_{i_2} \ldots z_{i_d} \in \mathcal{M}_d(Z)$, let A_m denote the matrix $A_{i_1} A_{i_2} \ldots A_{i_d}$. We see that

$$\begin{aligned} &f(A_1 z_1, A_2 z_2, \ldots, A_n z_n) \\ &\quad = \sum_{i_1, i_2, \ldots, i_d \in [n]} f[z_{i_1} z_{i_2} \cdots z_{i_d}](A_{i_1} z_{i_1})(A_{i_2} z_{i_2}) \cdots (A_{i_d} z_{i_d}) \\ &\quad = \sum_{i_1, i_2, \ldots, i_d \in [n]} f[z_{i_1} z_{i_2} \cdots z_{i_d}](A_{i_1} A_{i_2} \cdots A_{i_d})(z_{i_1} z_{i_2} \cdots z_{i_d}) \\ &\quad = \sum_{m \in \mathcal{M}_d(Z)} f[m] A_m m. \end{aligned}$$

Note that the coefficient $g[m]$ of a monomial $m = z_{i_1} z_{i_2} \cdots z_{i_d}$ in g is just $A_m(1, S) = \sum_{k_1, k_2, \ldots, k_{d-1} \in [S]} \prod_{j=1}^{d} A_{i_j}(k_{j-1}, k_j)$, where

$k_0 = 1$ and $k_d = S$. Putting the above observations together, we see that $f(A_1 z_1, A_2 z_2, \ldots, A_n z_n)(1, S) = \sum_{m \in \mathcal{M}_d(Z)} f[m] A_m(1, S) m = \sum_{m \in \mathcal{M}_d(Z)} f[m] g[m] m = f \circ g$. Since the entries of the matrices $A_1, A_2, \ldots, A_n$ can be read off from the labels of P, it can be seen that $A_1, A_2, \ldots, A_n$ can be computed in polynomial time given the ABP P. This completes the proof. □

Remark 3.3. *We note that the matrices A_i in the statement of Theorem 3.2 can actually be computed from the ABP even more efficiently, say, in uniform* AC^0.

The following corollary is immediate.

Corollary 3.4 (Arvind *et al.* 2009). *Given a noncommutative circuit of size S' for $f \in \mathbb{F}\langle Z \rangle$ and an ABP of size S for $g \in \mathbb{F}\langle Z \rangle$, we can efficiently compute a noncommutative circuit of size $O(S'S^3)$ for $f \circ g$.*

The next corollary is also quite useful for this paper.

Corollary 3.5. *Let $Z = \{z_1, \ldots, z_n\}$. Suppose $\mathcal{A}$ is a polynomial-time algorithm for computing a homogeneous degree d polynomial $f \in \mathbb{F}\langle Z \rangle$ for matrix inputs from $M_S(\mathbb{F})$. Given as input an ABP P, with S nodes, computing a homogeneous degree d polynomial $g \in \mathbb{F}\langle Z \rangle$, and scalars $a_1, a_2, \ldots, a_n \in \mathbb{F}$, we can compute $f \circ g(a_1, a_2, \ldots, a_n)$ in polynomial time.*

Proof (Corollary 3.5). We first compute matrices $A_1, A_2, \ldots, A_n$, described in the Theorem 3.2, in time polynomial in the description of the ABP P. Then we invoke the given algorithm $\mathcal{A}$ on input $(A_1 a_1, \ldots, A_n a_n)$ to obtain as output an $S \times S$ matrix whose $(1, S)^{th}$ entry contains $f \circ g(a_1, a_2, \ldots, a_n)$. Clearly, the simulation runs in polynomial time. □

Remark 3.6. *The statement can be generalized to any unital algebra in place of the field $\mathbb{F}$.*

4. The hardness of the Cayley determinant

We consider polynomials over an arbitrary field $\mathbb{F}$ (for the algorithmic results $\mathbb{F}$ is either $\mathbb{Q}$ or a finite field). The main result of this section is that if there is a polynomial-time algorithm to compute the $2n \times 2n$ Cayley determinant over inputs from $M_S(\mathbb{F})$ for $S = c \cdot n^2$ (for a suitable constant c) then there is a polynomial-time algorithm to compute the $n \times n$ permanent over $\mathbb{F}$.

Throughout this section let X denote $\{x_{ij} \mid 1 \leq i, j \leq 2n\}$, and Y denote $\{y_{ij} \mid 1 \leq i, j \leq n\}$. Our aim is to show that if there is a polynomial-time algorithm for computing $\mathrm{Cdet}_{2n}(X)$ where x_{ij} takes values in $M_S(\mathbb{F})$ then there is a polynomial-time algorithm which, on being given $y_{ij} \in \mathbb{F}$ $(i, j \in [n])$ as input, computes $\mathrm{Cperm}_n(Y)$ (which is the same as the standard permanent in the commutative setting).

The $2n \times 2n$ determinant has $(2n)!$ many signed monomials of degree $2n$ of the form $x_{1,\sigma(1)}x_{2,\sigma(2)} \cdots x_{2n,\sigma(2n)}$ for $\sigma \in S_{2n}$. We will identify $n!$ of these monomials, all of which have the same sign. More precisely, we will design a small ABP with which we will be able to pick out these $n!$ monomials of the same sign.

We now define these $n!$ many permutations from S_{2n} which have the same sign and the corresponding monomials of Cdet_{2n} that can be picked out by a small ABP.

Definition 4.1. *Let $n \in \mathbb{N}$. For each permutation $\pi \in S_n$, we define a permutation $\rho(\pi)$ in S_{2n}, called the* interleaving *of π, as follows:*

$$\rho(\pi)(i) = \begin{cases} \pi(\frac{i+1}{2}), & \text{if } i \text{ is odd,} \\ n + \pi(\frac{i}{2}), & \text{if } i \text{ is even.} \end{cases}$$

That is, the elements $\rho(\pi)(1), \rho(\pi)(2), \cdots, \rho(\pi)(2n)$ are simply $\pi(1), (n + \pi(1)), \pi(2), (n + \pi(2)), \cdots, \pi(n), (n + \pi(n))$.

The following lemma states a crucial property of the permutation $\rho(\pi)$.

Lemma 4.2. *The sign of the permutation $\rho(\pi)$ is independent of π. More precisely, for every $\pi \in S_n$, we have $\mathrm{sgn}(\rho(\pi)) = \mathrm{sgn}(\rho(1_n))$, where 1_n denotes the identity permutation in S_n.*

PROOF (Lemma 4.2). For each $\pi \in S_n$ we can define the permutation $\pi_2 \in S_{2n}$ as $\pi_2(i) = \pi(i)$ for $1 \leq i \leq n$ and $\pi_2(n+j) = n + \pi(j)$ for $1 \leq j \leq n$. It is easy to verify that $\mathrm{sgn}(\pi_2) = \mathrm{sgn}(\pi)^2 = 1$ for every $\pi \in S_n$. To see this we write π_2 as a product of disjoint cycles and notice that every cycle size occurs an even number of times. Furthermore, we can check that $\rho(\pi) = \rho(1_n)\pi_2$, where we evaluate products of permutations as we do for general functions, i.e. from right to left. Hence it follows that $\mathrm{sgn}(\rho(\pi)) = \mathrm{sgn}(\rho(1_n))\mathrm{sgn}(\pi_2) = \mathrm{sgn}(\rho(1_n))$. □

We will denote by ρ_0 the permutation $\rho(1_n)$, where 1_n denotes the identity permutation in S_n.

For $\sigma \in S_{2n}$, we denote by m_σ the monomial $x_{1,\sigma(1)} \cdots x_{2n,\sigma(2n)} \in \mathcal{M}(X)$. For $\sigma, \tau \in S_{2n}$, we denote $x_{\sigma(1),\tau(1)} \cdots x_{\sigma(2n),\tau(2n)}$ by $m_{\sigma,\tau}$.

In the next lemma, we show that there is an ABP that will filter out monomials that are not of the form $m_{\rho(\pi)}$ from among the m_σ.

LEMMA 4.3. *There is an ABP P of size $O(n^2)$ and width n that computes a homogeneous polynomial $F \in \mathbb{F}\langle X \rangle$ of degree $2n$ such that for any $\sigma, \tau \in S_{2n}$,*

- *$F[m_\sigma] = 1$ if $\sigma = \rho(\pi)$ for some $\pi \in S_n$, and 0 otherwise.*
- *$F[m_{\sigma,\tau}] = 0$ unless $\sigma = 1_{2n}$, where 1_{2n} denotes the identity permutation in S_{2n}.*

Moreover, the above ABP P can be computed in time $\mathrm{poly}(n)$.

PROOF (Lemma 4.3). The ABP is essentially just a finite automaton over the alphabet X with the following properties: for input monomials of the form m_σ it accepts only those monomials that are of the form $m_{\rho(\pi)}$. Further, for input monomials of the form $m_{\sigma,\tau}$ it accepts only those monomials of the form $m_{1_{2n},\tau}$. We give the formal description of this ABP P below.

The ABP P contains $2n+1$ layers, labelled $\{0, 1, \ldots, 2n\}$. For each even $i \in \{0, 1, \ldots, 2n\}$, there is exactly one node q_i at level i; for each odd $i \in \{0, 1, \ldots, 2n\}$, there are n nodes $p_{i,1}, p_{i,2}, \ldots, p_{i,n}$ at level i. We now describe the edges of P: for each even $i \in \{0, 1, \ldots, 2n-2\}$ and $j \in [n]$, there is an edge from q_i to $p_{i+1,j}$

labelled $x_{i+1,j}$; for each odd $i \in \{0, 1, \ldots, 2n\}$ and $j \in [n]$, there is an edge from $p_{i,j}$ to q_{i+1} labelled $x_{i+1,n+j}$.

It is easy to see that P as defined above satisfies the requirements of the statement of the lemma. It is also clear that the ABP P can be computed in polynomial time. □

Note that the ABP P of Lemma 4.3 can in fact be constructed in uniform AC^0.

Remark 4.4. *For this section, we require only the first part of Lemma 4.3. The second part of Lemma 4.3 is used in Section 6.*

We are now ready to prove that if there is a small noncommutative arithmetic circuit that computes the Cayley determinant polynomial, then there is a small noncommutative arithmetic circuit that computes the Cayley permanent polynomial.

Theorem 4.5. *For any $n \in \mathbb{N}$, if there is a circuit C of size s computing $\mathrm{Cdet}_{2n}(X)$, then there is a circuit C' of size polynomial in s and n that computes $\mathrm{Cperm}_n(Y)$.*

Proof (Theorem 4.5). Assuming the existence of the circuit C as stated above, by Corollary 3.4, there is a noncommutative arithmetic circuit C'' of size $\mathrm{poly}(s, n)$ that computes the polynomial $F'' = \mathrm{Cdet}_{2n} \circ F$, where F is the polynomial referred to in Lemma 4.3. For any monomial m, if $m \neq m_\sigma$ for any $\sigma \in S_{2n}$, then $\mathrm{Cdet}_{2n}[m] = 0$ and hence, in this case, $F''[m] = 0$; moreover, for $m = m_\sigma$, we have $F[m] = 1$ if $\sigma = \rho(\pi)$ for some $\pi \in S_n$, and 0 otherwise. Hence, we see that

$$F''(X) = \sum_{\pi \in S_n} \mathrm{sgn}(\rho(\pi)) m_{\rho(\pi)} = \mathrm{sgn}(\rho_0) \left(\sum_{\pi \in S_n} m_{\rho(\pi)} \right)$$

where the last equality follows from Lemma 4.2.

Let C' be the circuit obtained from C'' by substituting x_{ij} with $y_{\frac{1+i}{2},j}$ if i is odd and $j \in [n]$, and by 1 if i is even or $j \notin [n]$, and by multiplying the output of the resulting circuit by $\mathrm{sgn}(\rho_0)$. Let F' denote the polynomial computed by C'. Then, we have

$$F'(X) = \sum_{\pi \in S_n} m'_{\rho(\pi)}$$

where $m'_{\rho(\pi)}$ denotes the monomial obtained from $m_{\rho(\pi)}$ after the substitution. It can be checked that for any $\pi \in S_n$, the monomial $m'_{\rho(\pi)} = y_{1,\pi(1)} y_{2,\pi(2)} \cdots y_{n,\pi(n)}$. Hence, the polynomial F' computed by C' in indeed $\mathrm{Cperm}_n(Y)$. It is easily seen that the size of C' is $\mathrm{poly}(s, n)$. □

We now show that evaluating the polynomial Cdet_{2n} over $M_S(\mathbb{F})$, for $S = c \cdot n^2$ for suitable $c > 0$, is at least as hard as evaluating the permanent over $\mathbb{F}$.

THEOREM 4.6. *If there is a polynomial-time algorithm $\mathcal{A}$ that computes the $2n \times 2n$ Cayley determinant of matrices with entries in $M_S(\mathbb{F})$, for $S = c \cdot n^2$ for suitable $c > 0$, then there is a polynomial-time algorithm that computes the $n \times n$ permanent over $\mathbb{F}$.*

PROOF (Theorem 4.6). This is an easy consequence of Corollary 3.5. Consider the algorithm given by Corollary 3.5 for computing $\mathrm{Cdet}_{2n} \circ F$ over the field $\mathbb{F}$, where the ABP in Corollary 3.5 is the ABP of Lemma 4.3 computing F.

In order to evaluate the permanent over inputs a_{ij} $(1 \le i, j \le n)$ we will substitute $x_{2i-1,j} = a_{ij}$ for $1 \le i, j \le n$ and we put $x_{i,j} = 1$ when i is even or $j > n$. As in the proof of Theorem 4.5 it follows that for this substitution the algorithm computing $\mathrm{Cdet}_{2n} \circ F$ will output $\mathrm{sgn}(\rho_0)\mathrm{Cperm}_n(a_{11}, \ldots, a_{nn})$. Since $\mathrm{sgn}(\rho_0)$ can be easily computed, we have a polynomial-time algorithm for computing the $n \times n$ permanent over $\mathbb{F}$. □

REMARK 4.7. *The above result has a stronger consequence: for any fixed $\varepsilon > 0$, if there is a polynomial-time algorithm that computes the $m \times m$ Cayley determinant over $M_{m^\varepsilon}(\mathbb{F})$, then there is a polynomial-time algorithm that computes $\Omega(m^{\varepsilon/2}) \times \Omega(m^{\varepsilon/2})$ permanents over $\mathbb{F}$, hence implying that permanent over $\mathbb{F}$ is polynomial-time computable.*

5. The Cayley determinant over Clifford algebras

We now consider the complexity of computing the determinant over real Clifford algebras of polynomially large dimension. We show via a polynomial-time reduction that computing the permanent over the rationals is reducible to this problem. Indeed, by inspecting our result we can observe that even *approximating* the determinant over such Clifford algebras would yield similar approximation algorithms for the permanent over the reals.

We first define the basic notions in the theory of Clifford algebras. A more thorough treatment can be found in Chien *et al.* (2003) and the references therein. Fix $m \in \mathbb{N}$. The (real) *Clifford algebra* CL'_m is a 2^m-dimensional vector space over $\mathbb{R}$ with basis elements of the form $e_{i_1}e_{i_2}e_{i_3}\cdots e_{i_k}$ where $i_1 < i_2 < i_3 \cdots < i_k$ are elements from $[m]$. Multiplication between elements of the basis is defined by the following rules: $e_i^2 = 1$ and $e_ie_j = -e_je_i$ for distinct $i, j \in [m]$; this is extended linearly to all pairs of elements from the Clifford algebra. Given $i_1 < i_2 < \cdots < i_k$ from $[m]$, we denote by e_S the basis element $e_{i_1}e_{i_2}\cdots e_{i_k}$, where $S = \{i_1, i_2, \ldots, i_k\}$. Each element of the Clifford algebra is uniquely expressible as $\sum_{S\subseteq[m]} c_S e_S$, where $c_S \in \mathbb{R}$ for each S. (Note that $e_\emptyset$ and 1 both refer to the multiplicative identity of the algebra.) An *idempotent* of the Clifford algebra is an element e such that $e^2 = e$. Given $h = \sum_{S\subseteq[m]} c_S e_S$ in CL'_m, we define its *norm* $|h|$ to be $\sqrt{\sum_{S\subseteq[m]} c_S^2}$.

The subset of basis elements $\{e_S \mid |S| \text{ even}\}$ generates a strict subalgebra of CL'_m. We will denote this subalgebra by CL_m. This is the algebra of interest to us. The term 'Clifford algebra' will henceforth refer to CL_m for some $m \in \mathbb{N}$.

Chien, Rasmussen, and Sinclair (Chien *et al.* 2003) have shown that a polynomial-time algorithm that, when given as input an $n\times n$ matrix B with entries from CL_m for $m = 2\log n+2$, computes $|\mathrm{Cdet}_n(B)|^2$ can be used to design a randomized polynomial-time algorithm to approximate the 0-1 permanent (over $\mathbb{Q}$).

In this section, we prove that if there is a polynomial-time algorithm to compute either $|\mathrm{Cdet}_n(B)|^2$ or $\mathrm{Cdet}_n(B)$, then the permanent (over inputs from $\mathbb{Q}$) can actually be computed in polynomial

time. For an $n \times n$ real matrix A, let $\mathrm{perm}_n(A)$ denote the permanent of A.

Remark 5.1. *In a sense, our result in this section should not be surprising. We have already proved (in Theorem 4.6) that computing the determinant over matrix algebras is at least as hard as computing the permanent. Also, it is known that Clifford algebras of polynomial dimension are isomorphic to matrix algebras of polynomial dimension (see, for example, (Lam 2005, Chapter 5)). However, in this section we actually give an explicit polynomial-time reduction showing that computing the permanent over the reals is reducible to computing either* $|\mathrm{Cdet}_n(B)|^2$ *or* $\mathrm{Cdet}_n(B)$ *where the entries of* B *are from the Clifford algebra* CL_m.

Suppose we wish to compute the permanent of an $n \times n$ matrix with entries from $\mathbb{Q}$. W.l.o.g., we assume that $n = 2^\ell$ for some $\ell \in \mathbb{N}$. Let m denote 5ℓ. The next lemma is about the existence of certain elements in the algebra CL_m useful for the reduction.

Lemma 5.2. *Let* n, ℓ, m *be as above. Then, there exist* $h_1, h_2, \ldots, h_n, h'_1, h'_2, \ldots, h'_n \in CL_m$ *and an idempotent* $e \in CL_m$ *such that:*

- *For all* j, $h_j h'_j = e$.
- *For all* $j \neq k$, $h_j h'_k = 0$.
- $|e|^2 = \frac{1}{2^\ell}$.

Moreover, the elements $h_1, h_2, \ldots, h_n, h'_1, h'_2, \ldots, h'_n$ *and* e *can be constructed in time* $\mathrm{poly}(n)$.

We defer the proof of the above lemma and first prove the main result of this section.

Theorem 5.3. *Let* n, ℓ, m *be as above. There is a polynomial-time algorithm which, when given any matrix* $A \in M_n(\mathbb{R})$, *computes a* $B \in M_{2n}(CL_m)$ *such that* $|\mathrm{Cdet}_{2n}(B)|^2 = \frac{\mathrm{perm}_n(A)^2}{2^\ell}$.

Proof (Theorem 5.3). The matrix B will be the following: for any odd $i \in [2n]$ and any $j \in [2n]$, set $B(i, j)$—the (i, j)th entry of

B—to be $A(\frac{i+1}{2}, j)h_j$ if $j \leq n$ and 0 if $j > n$; for any even $i \in [2n]$ and any $j \in [2n]$, set $B(i,j)$ to be h'_{j-n} if $j > n$ and 0 otherwise. Clearly, B can be computed in polynomial time given A. Note the following property of B: for any odd $i \in [2n]$ and $j, k \in [2n]$

$$B(i,j)B(i+1,k) = \begin{cases} A(\frac{i+1}{2}, j)e \text{ if } j \leq n \text{ and } k = n+j, \\ 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Here e denotes the idempotent from Lemma 5.2. The following claim is easy to see.

Claim 5.4. *For any bijection $\sigma \in S_{2n}$, the product $\prod_{i=1}^{2n} B(i,\sigma(i))$ $= (\prod_{i=1}^{n} A(i,\pi(i)))e$ if $\sigma = \rho(\pi)$ for some $\pi \in S_n$ and it is 0 otherwise.*

Let us consider $\mathrm{Cdet}_{2n}(B)$. We have:

$$\begin{aligned} \mathrm{Cdet}_{2n}(B) &= \sum_{\sigma \in S_{2n}} \mathrm{sgn}(\sigma) B(1,\sigma(1)) \cdot B(2,\sigma(2)) \cdots B(2n,\sigma(2n)) \\ &= \sum_{\pi \in S_n} \mathrm{sgn}(\rho(\pi))(\prod_{i=1}^{n} A(i,\pi(i)))e \\ &= \mathrm{sgn}(\rho_0)\mathrm{perm}_n(A)e. \end{aligned}$$

Thus, we see that $|\mathrm{Cdet}_{2n}(B)|^2 = \mathrm{perm}_n(A)^2 |e|^2 = \frac{\mathrm{perm}_n(A)^2}{2^\ell}$. □

We have the following easy consequence of the above theorem.

Corollary 5.5. *Fix any $\varepsilon > 0$, and suppose there is a polynomial-time algorithm that computes $|\mathrm{Cdet}_n(B)|^2$ on input an $n \times n$ matrix B with entries from CL_m for $m = \varepsilon \log n$. Then there is a polynomial-time algorithm that computes the $n \times n$ permanent of matrices with nonnegative rational entries.*

Proof (Corollary 5.5). The statement directly follows from Theorem 5.3 for $m = \lceil 5 \log n \rceil$. To prove hardness for $m = \varepsilon \log n$, we note that a polynomial-time algorithm to compute $|\mathrm{Cdet}_n(B)|^2$ over $CL_{\varepsilon \log n}$ can be used to compute $|\mathrm{Cdet}_{n^{\varepsilon/5}}(B)|^2$ over $CL_{5 \log n^{\varepsilon/5}}$ in polynomial time. □

A δ-approximation algorithm $\mathcal{A}$ for a function $f : \Sigma^* \longrightarrow \mathbb{Q}$ is an algorithm such that for each $x \in \Sigma^*$

$$(1-\delta)f(x) \leq \mathcal{A}(x) \leq (1+\delta)f(x).$$

Our reduction from computing the permanent for nonnegative entries to computing $|\mathrm{Cdet}_n(B)|^2$ actually yields an approximation preserving reduction. We formalize this in the next corollary.

COROLLARY 5.6. *Fix any $\delta > 0$ and $\varepsilon > 0$. Suppose there is a polynomial-time δ-approximation algorithm for the function that on input an $n \times n$ matrix B with entries from CL_m for $m = \varepsilon \log n$ takes the value $|\mathrm{Cdet}_n(B)|^2$. Then there is a polynomial-time δ-approximation algorithm for the $n \times n$ permanent with nonnegative rational entries.*

We now prove Lemma 5.2.

PROOF (Lemma 5.2). Let $e_1, e_2, \ldots, e_m$ denote the generators of CL'_m. Partition the set $[m]$ into ℓ subsets of size 5 as follows: set $S_i = \{5(i-1)+j \mid j \in [5]\}$ for each $i \in [\ell]$. For each $i \in [\ell]$, let $S_{i,0} = \{5(i-1)+1, 5(i-1)+2, 5(i-1)+3, 5(i-1)+5\}$ and $S_{i,1} = \{5(i-1)+2, 5(i-1)+3, 5(i-1)+4, 5(i-1)+5\}$.

Using the fact that $e_i^2 = 1$ and $e_i e_j = -e_j e_i$ for $i \neq j$ it easily follows that for any two *disjoint* sets $S, T \subseteq [m]$ such that $|S|, |T|$ are even, we have $e_S e_T = e_T e_S$. Hence, the elements $e_{S_{i,b_1}}$ and $e_{S_{j,b_2}}$ commute for $i \neq j$ and any $b_1, b_2 \in \{0,1\}$. Furthermore, for all $i \in [\ell]$ and $b \in \{0,1\}$ we have $e^2_{S_{i,b}} = 1$. Also, we have $e_{S_{i,0}} e_{S_{i,1}} = -e_{S_{i,1}} e_{S_{i,0}}$. Finally, notice that $e_{S_{i,b}}$ for $1 \leq i \leq \ell$ and $b \in \{0,1\}$ are all elements of CL_m.

For $i \in [\ell]$ and $b \in \{0,1\}$, set $g_{i,0} = \frac{1+e_{S_{i,1}}}{2}$ and $g_{i,1} = \frac{e_{S_{i,0}}(1-e_{S_{i,1}})}{2}$. Also, set $g'_{i,0} = g_{i,0}$ and $g'_{i,1} = \frac{e_{S_{i,0}}(1+e_{S_{i,1}})}{2}$. Notice that $g^2_{i,0} = g_{i,0}$. We also note an additional relation $e_{S_{i,0}}(1-e_{S_{i,1}}) = (1+e_{S_{i,1}})e_{S_{i,0}}$. Using these we can easily derive the following crucial properties of these elements of CL_m.

- For each $i \in [\ell]$ and $b \in \{0,1\}$, $g_{i,b} g'_{i,b} = g_{i,0}$.
- For each $i \in [\ell]$ and $b \in \{0,1\}$, $g_{i,b} g'_{i,1-b} = 0$.

- For $i_1 \neq i_2$ and any $b_1, b_2 \in \{0,1\}$, the elements g_{i_1,b_1} and g'_{i_2,b_2} commute.

Finally, we define h_j, h'_j for a fixed $j \in [n]$. Let $b_1 b_2 \ldots b_\ell$ be the binary representation of the integer $j-1$ (recall that $n = 2^\ell$). We define $h_j = g_{1,b_1} g_{2,b_2} \cdots g_{\ell,b_\ell}$ and $h'_j = g'_{1,b_1} g'_{2,b_2} \cdots g'_{\ell,b_\ell}$. Also, we define e to be $g_{1,0} g_{2,0} \cdots g_{\ell,0}$, which is the same as h_1 and h'_1.

We now prove that the h_j, h'_j ($j \in [n]$) and e satisfy the properties claimed in the statement of the lemma. Fix any $j \in [n]$ and let $b_1 b_2 \ldots b_\ell$ be the binary representation of $j-1$. We have

$$\begin{aligned} h_j h'_j &= g_{1,b_1} g_{2,b_2} \cdots g_{\ell,b_\ell} g'_{1,b_1} g'_{2,b_2} \cdots g'_{\ell,b_\ell} \\ &= (g_{1,b_1} g'_{1,b_1}) \cdot (g_{2,b_2} g'_{2,b_2}) \cdots (g_{\ell,b_\ell} g'_{\ell,b_\ell}) \\ &= g_{1,0} g_{2,0} \cdots g_{\ell,0} = e \end{aligned}$$

The second equality follows from the fact that $g_{i_1,b}$ and $g'_{i_2,b}$ commute for any distinct i_1 and i_2. The third equality follows from the fact that for any i and b, $g_{i,b} g'_{i,b} = g_{i,0}$. This proves the first property claimed in the statement of the lemma. Similarly, we can see that e is an idempotent: $e^2 = h_1^2 = e$.

Fix any distinct $j, k \in [n]$. Let $b_1 b_2 \ldots b_\ell$ and $b'_1 b'_2 \ldots b'_\ell$ be the binary representations of j and k. Since $j \neq k$, we can fix some i such that $b'_i = 1 - b_i$. We have

$$\begin{aligned} h_j h'_k &= g_{1,b_1} g_{2,b_2} \cdots g_{\ell,b_\ell} g'_{1,b'_1} g'_{2,b'_2} \cdots g'_{\ell,b'_\ell} \\ &= (g_{1,b_1} g'_{1,b_1}) \cdot (g_{2,b_2} g'_{2,b_2}) \cdots (g_{i,b_i} g'_{i,b'_i}) \cdots (g_{\ell,b_\ell} g'_{\ell,b_\ell}) \\ &= (g_{1,b_1} g'_{1,b_1}) \cdot (g_{2,b_2} g'_{2,b_2}) \cdots 0 \cdots (g_{\ell,b_\ell} g'_{\ell,b_\ell}) = 0 \end{aligned}$$

where the third equality follows from the fact that we have $g_{i,b} g'_{i,1-b} = 0$. This proves the second claim made in the lemma.

Finally, we note that

$$\begin{aligned} |e|^2 = |g_{1,0} g_{2,0} \cdots g_{\ell,0}|^2 &= \left| \frac{1}{2^\ell} \sum_{T \subseteq \ell} \prod_{i \in T} e_{S_{i,1}} \right|^2 \\ = \frac{1}{4^\ell} \left| \sum_{T \subseteq \ell} \prod_{i \in T} e_{S_{i,1}} \right|^2 &= \frac{2^\ell}{4^\ell} = \frac{1}{2^\ell} \end{aligned}$$

It is easily seen from their definitions that the h_j, h'_j and e can be computed in time poly(n). This completes the proof of the lemma. □

6. The symmetrized determinant

In this section, we observe that the $2n \times 2n$ symmetrized determinant over $O(n^2)$-dimensional matrix algebras is at least as hard to compute as the permanent. This may be compared to the result of Barvinok (2000), who shows that over *constant-dimensional* matrix algebras, the symmetrized determinant is polynomial-time computable.

In this section, let $\mathbb{F}$ denote a field of characteristic 0. Let $X = \{x_{ij} \mid 1 \le i, j \le 2n\}$ and $Y = \{y_{ij} \mid 1 \le i, j \le n\}$. Recall that for $\sigma, \tau \in S_{2n}$, the monomial $m_{\sigma,\tau}$ is $x_{\sigma(1),\tau(1)}x_{\sigma(2),\tau(2)}\cdots x_{\sigma(2n),\tau(2n)}$, and the monomial m_σ is $x_{1,\sigma(1)}x_{2,\sigma(2)}\cdots x_{2n,\sigma(2n)}$.

THEOREM 6.1. *If the* $\mathrm{sdet}_{2n}(X)$ *polynomial over* $\mathbb{F}$ *can be computed by a polynomial-sized noncommutative arithmetic circuit, then the polynomial* $\mathrm{Cperm}_n(Y)$ *can be computed by a polynomial-sized noncommutative arithmetic circuit.*

PROOF (Theorem 6.1). Assume that $\mathrm{sdet}_{2n}(X)$ is computed by a circuit C of size s. As in Theorem 4.5, we will proceed by taking Hadamard product. Let P be the ABP defined in Lemma 4.3 and $F(X)$ the polynomial it computes. Let F'' denote the polynomial $\mathrm{sdet}_{2n}(X) \circ F$. Note that by Corollary 3.4, F'' can be computed by a circuit C'' of size poly(s, n). From Lemma 4.3, we have $F[m_{\sigma,\tau}] = 0$ unless $\sigma = 1_{2n}$, the identity permutation in S_{2n}; moreover, we also have $F[m_{1_{2n},\tau}] = F[m_\tau]$ which is 1 if $\tau = \rho(\pi)$ for some $\pi \in S_n$ and 0 otherwise. By the above reasoning,

$$F''(X) = \frac{1}{(2n)!}\sum_{\pi\in S_n}\mathrm{sgn}(\rho(\pi))m_{\rho(\pi)} = \frac{\mathrm{sgn}(\rho_0)}{(2n)!}\sum_{\pi\in S_n} m_{\rho(\pi)}$$

where $\rho_0 = \rho(1_n)$ and 1_n the identity permutation in S_n.

Now, we substitute each x_{ij} by $y_{\frac{1+i}{2},j}$ if i is odd and $j \in [n]$ and by 1 if i is even or $j \notin [n]$ in the circuit C''. The effect of this substitution is to transform $m_{\rho(\pi)}$ into $y_{1,\pi(1)}y_{2,\pi(2)}\cdots y_{n,\pi(n)}$ for each

$\pi \in S_n$. Hence, the resulting polynomial is simply $\frac{\mathrm{sgn}(\rho_0)\mathrm{Cperm}_n(Y)}{(2n)!}$. Thus, by multiplying by $\mathrm{sgn}(\rho_0)(2n)!$, we obtain a circuit C' of size $\mathrm{poly}(s, n)$ that computes $\mathrm{Cperm}_n(Y)$. □

THEOREM 6.2. *If there is a polynomial-time algorithm $\mathcal{A}$ that computes the $2n \times 2n$ symmetrized determinant of matrices with entries in $M_S(\mathbb{F})$, for $S = c \cdot n^2$ for suitable $c > 0$, then there is a polynomial-time algorithm that computes the $n \times n$ permanent over $\mathbb{F}$.*

PROOF (Theorem 6.2). The proof is almost exactly identical to that of Theorem 4.6. Consider the algorithm given by Corollary 3.5 for computing $\mathrm{sdet}_{2n} \circ F$ over the field $\mathbb{F}$, where the ABP in Corollary 3.5 is the ABP of Lemma 4.3 computing F.

In order to evaluate the permanent over inputs $a_{ij}, 1 \leq i, j \leq n$ we will substitute $x_{2i-1,j} = a_{ij}$ for $1 \leq i, j \leq n$ and we put $x_{i,j} = 1$ when i is even or $j > n$. As in the proof of Theorem 6.1, it follows that for this substitution the algorithm computing $\mathrm{sdet}_{2n} \circ F$ will output $\frac{\mathrm{sgn}(\rho_0)}{(2n)!}\mathrm{Cperm}_n(a_{11}, \ldots, a_{nn})$. Since $\mathrm{sgn}(\rho_0)$ and $(2n)!$ are easily computable, we have a polynomial-time algorithm for computing the $n \times n$ permanent over $\mathbb{F}$. □

7. The Moore determinant

We demonstrate by a simple reduction that the Moore determinant and permanent are inter-reducible. We also show that the computing the Moore determinant over a field of characteristic zero is at least as hard as counting the number of directed Hamilton Cycles of a directed graph, which is a well-known #P-complete problem. If the field is of characteristic k, then computing the Moore determinant over the field is at least as hard as counting the number of Hamilton cycles of a directed graph modulo the prime k, which is hard for $\mathrm{Mod}_k\mathrm{P}$.

Assume $X = \{x_{ij} \mid 1 \leq i, j \leq n\}$. Given a permutation $\sigma \in S_n$, we write σ as a product of $r = r(\sigma)$ disjoint cycles as follows: we write $\sigma = (n^\sigma_{11} \cdots n^\sigma_{1l_1})(n^\sigma_{21} \cdots n^\sigma_{2l_2}) \cdots (n^\sigma_{r1} \cdots n^\sigma_{rl_r})$ with $n^\sigma_{i1} < n^\sigma_{ij}$ for all $i \in [r]$ and $j \in [l_i] \setminus \{1\}$ and satisfying $n^\sigma_{11} > n^\sigma_{21} > \cdots >$

n^σ_{r1}. Let w_σ denote the monomial $x_{n^\sigma_{11},n^\sigma_{12}} \cdots x_{n^\sigma_{1l_r},n^\sigma_{11}} \cdots x_{n^\sigma_{r1},n^\sigma_{r2}} \cdots x_{n^\sigma_{rl_r},n^\sigma_{r1}}$.

Let C_n denote the set of all 1-cycles in S_n, i.e permutations whose cycle decomposition consists of a single cycle of length n. Define the polynomial $\mathrm{HC}_n(x_{11}, \ldots, x_{nn}) \in \mathbb{F}\langle X\rangle$ to be $\sum_{\sigma\in C_n} w_\sigma$. Fix any directed graph G on n vertices with adjacency matrix A. Let $H(G)$ denote $\mathrm{HC}_n(A(1,1), \ldots, A(n,n))$. The quantity $H(G)$ has a simple description: if $\mathbb{F}$ is of characteristic 0, then $H(G)$ is the number of directed Hamiltonian cycles in G; and if $\mathbb{F}$ is of characteristic $k > 0$, then $H(G)$ is the number of directed Hamiltonian cycles of G modulo k.

We have the following easy lemma:

LEMMA 7.1. *There are ABPs P'_1 and P'_2 of size $O(n^2)$ and width n that compute homogeneous polynomials $F'_1, F'_2 \in \mathbb{F}\langle X\rangle$ of degree n such that for any $\sigma \in S_n$, we have*

- $F'_1[w_\sigma] = \mathrm{sgn}(\sigma)$.
- $F'_2[w_\sigma] = \mathrm{sgn}(\sigma)$ *if* $\sigma \in C_n$ *and* 0 *otherwise.*

Moreover, the above ABPs can be computed in time $\mathrm{poly}(n)$.

PROOF (Lemma 7.1). Recall that given a permutation $\sigma \in S_n$, the quantity $\mathrm{sgn}(\sigma)$ is $(-1)^{n+c_\sigma}$, where c_σ is the number of cycles in σ. Moreover, note that if σ as a product of disjoint cycles is

$$(n^\sigma_{11} \cdots n^\sigma_{1l_1})(n^\sigma_{21} \cdots n^\sigma_{2l_2}) \cdots (n^\sigma_{r1} \cdots n^\sigma_{rl_r})$$

as above, the value c_σ is simply the number of *left-to-right minima* in this representation, i.e the number of n^σ_{ij} such that $n^\sigma_{ij} < n^\sigma_{kl}$ for all n^σ_{kl} to the *left* of n^σ_{ij}.

Using this observation, it is easy to design an ABP P'_1 that keeps track of the sign of the permutation and the last left-to-right minimum seen, and computes a polynomial F'_1 as above. This can be done by maintaining at each layer $d \in \{0, 1, \ldots, n-1\}$, a vertex for each pair (m_1, m_2) of integers from $\{0, \ldots, n\}$—the vertex is named (d, m_1, m_2)—where m_1 is the least i such that a variable of the form $x_{i,j}$ has been encountered along the path from the source vertex $(0, 0, 0)$ on layer 0 to (d, m_1, m_2) and m_2 is the number of

left-to-right minima seen so far along this path. Each outgoing edge is now labelled by a variable from X multiplied by ± 1. If this variable is $x_{i,j}$ where $i \geq m_1$, then the edge is routed to vertex $(d+1, m_1, m_2)$.[3] If this variable is $x_{i,j}$ where $i < m_1$, then the edge is labelled with $-x_{i,j}$ and routed to $(d+1, i, m_2+1)$. All the vertices on layer n are identified and become the sink node of the ABP. This ABP performs exactly as intended except that the polynomial F_1'' computed satisfies $F_1''[w_\sigma] = (-1)^n \mathrm{sgn}(\sigma)$. It is easy to now produce P_1' as necessary by multiplying all the edges between (say) the first two layers with $(-1)^n$.

The ABP P_2' can be constructed similarly; the main difference from the case of P_1' is that the ABP must produce the coefficient 0 unless $n_{11}^\sigma = 1$. We omit the formal descriptions of the ABPs. □

The analogue of Theorem 4.5 for the Moore determinant follows below. The statement here is stronger: we show that the arithmetic circuit complexity of $\mathrm{Mdet}_n(X)$ is polynomial *if and only if* the arithmetic circuit complexity of $\mathrm{Mperm}_n(X)$ is polynomial.

THEOREM 7.2. *The Moore determinant polynomial* $\mathrm{Mdet}_n(X)$ *can be computed by a polynomial-sized noncommutative arithmetic circuit if and only if the Moore permanent polynomial* $\mathrm{Mperm}_n(X)$ *can be computed by a polynomial-sized noncommutative arithmetic circuit.*

PROOF (Theorem 7.2). As in the proof of Theorem 4.5, we will use the Hadamard product; this time, it can be used to erase or introduce the signs of the permutations corresponding to each monomial w_σ. Formally, we have $\mathrm{Mperm}_n(X) = \mathrm{Mdet}_n(X) \circ F_1'(X)$ and $\mathrm{Mdet}_n(X) = \mathrm{Mperm}_n(X) \circ F_1'(X)$, where $F_1'(X)$ is the polynomial defined in the statement of Lemma 7.1. Hence, if $\mathrm{Mdet}_n(X)$ (resp. $\mathrm{Mperm}_n(X)$) is computed by a noncommutative arithmetic circuit of size s, then by applying Corollary 3.4, we see that $\mathrm{Mperm}_n(X)$ (resp. $\mathrm{Mdet}_n(X)$) is computed by a noncommutative arithmetic circuit of size $\mathrm{poly}(s, n)$. □

[3] Strictly speaking, the case $i = m_1$ is irrelevant since in each w_σ for each i, only one variable of the form $x_{i,j}$ can occur.

Remark 7.3. *Note that Theorem 7.2 proves an equivalence (up to polynomial factors) between the arithmetic circuit complexities of the Moore determinant and permanent. This is a stronger statement than we obtained in the case of the Cayley determinant and permanent, where we only showed (roughly) that the Cayley determinant is at least as hard to compute as the Cayley permanent. The reason for this is that we are unable to obtain a small ABP that performs the function of P_1' for the monomials m_σ (defined in Section 4): that is, a small ABP computing a polynomial F_1 such that $F_1[m_\sigma] = \mathrm{sgn}(m_\sigma)$ for every $\sigma \in S_n$. Nevertheless, such a converse can be obtained by appealing to the result of Hrubeš et al. (Hrubeš et al. 2010) who show that the Cayley Permanent is complete for the complexity class VNP in the noncommutative setting.*

We now consider the complexity of computing the Moore determinant over matrix algebras of polynomial dimension. We can, as in the previous sections, show that this is at least as hard as computing the permanent over matrices with entries from $\mathbb{F}$, but we take a different route this time. We show that if the Moore determinant over a field of characteristic k can be computed in polynomial time, then there is a polynomial-time algorithm to compute the number of directed Hamilton cycles $H(G)$ modulo k for an input directed graph G. This allows us to draw stronger consequences, namely that the Moore determinant is hard to compute even when the field $\mathbb{F}$ is of characteristic 2, something that would not follow if we reduced the permanent to this problem (since the permanent is polynomial-time computable over fields of characteristic 2).

Theorem 7.4. *If there is a polynomial-time algorithm $\mathcal{A}$ that computes the $n \times n$ Moore determinant of matrices with entries in $M_S(\mathbb{F})$, for $S = c \cdot n^2$ for suitable $c > 0$, then there is a polynomial-time algorithm that, on input a directed graph G, computes $H(G)$.*

Proof (Theorem 7.4). Note that $\mathrm{HC}_n(X) = \mathrm{Mdet}_n(X) \circ F_2'$, where F_2' is the polynomial computed by ABP P_2' constructed in Lemma 7.1. Moreover, $H(G) = \mathrm{HC}_n(A(1,1), \ldots, A(n,n))$, where A is the adjacency matrix of the graph G. Hence, to compute

$H(G)$, we need to compute $\mathrm{HC}_n(A(1,1),\ldots,A(n,n))$, which can be done in polynomial time by Corollary 3.5. □

8. Completeness results

In this section, we observe that the noncommutative Cayley determinant over integer matrices is complete for GapP w.r.t. polynomial-time Turing reductions. Likewise, the noncommutative Cayley determinant over a finite field of characteristic $k \neq 2$ is hard for the modular counting complexity class $\mathrm{Mod}_k\mathrm{P}$ w.r.t. polynomial-time Turing reductions. These observations also hold for the symmetrized determinant. For the Moore determinant, we prove the above results without any restriction on the characteristic of the underlying field. We formally describe these observations.

Definition 8.1 (Beigel & Gill 1992; Fenner *et al.* 1994). *A function $f : \Sigma^* \longrightarrow \mathbb{Z}$ is in* GapP *if there is a polynomial-time NDTM M such that for each $x \in \Sigma^*$ the value $f(x)$ is $\mathrm{acc}_M(x) - \mathrm{rej}_M(x)$.*

For a prime k, the class $\mathrm{Mod}_k\mathrm{P}$ *consist of languages $L \subseteq \Sigma^*$ such that for some function $f \in$ GapP we have $x \in L$ if and only if $f(x) \equiv 0 (mod\ k)$.*

By Valiant's result (Valiant 1979) it is known that the integer permanent is GapP-complete with respect to polynomial-time Turing reductions. Furthermore, the permanent over $\mathbb{F}_k$ is Mod_kP-hard for prime $k \neq 2$.

Now, for $n \in \mathbb{N}$, consider the Cayley determinant for $2n \times 2n$ matrices with entries from $M_S(\mathbb{Z})$, where $S = cn^2$ for some constant c. By Theorem 4.6, there is a fixed $c > 0$ such that computing the integer permanent for $n \times n$ matrices is polynomial-time reducible to computing the $(1,S)^{th}$ entry of such a Cayley determinant. The same observation holds modulo k for a prime k.

Furthermore, the problem of computing the $(1,S)^{th}$ entry of such a Cayley determinant over $\mathbb{Z}$ is easily seen to be in GapP: we can design a polynomial-time NDTM which takes as input a $2n \times 2n$ matrix with entries from $M_S(\mathbb{Z})$ and the difference in the number of accepting and rejecting paths is the $(1,S)^{th}$ entry of its Cayley determinant. A similar argument, using Theorem 6.2,

works for the Symmetrized determinant as well. Hence we have the following.

COROLLARY 8.2. *There exists a constant c such that the following holds. For $S = cn^2$, computing the $(1,S)^{th}$ entry of the Cayley determinant or the Symmetrized determinant for $2n \times 2n$ matrices with entries from $M_S(\mathbb{Z})$ is* GapP*-complete w.r.t. polynomial-time Turing reductions. Further, given a finite field $\mathbb{F}$ of characteristic $k \neq 2$, computing the $(1,S)^{th}$ of the Cayley determinant for $2n \times 2n$ matrices over $M_S(\mathbb{F})$ is hard w.r.t. polynomial-time Turing reductions for* $\mathrm{Mod}_k\mathrm{P}$.

REMARK 8.3. *As briefly remarked in the introduction, the above result for the* Cayley *determinant has been superseded by results of Bläser (2015); Chien et al. (2011); Gentry (2014). The work of Bläser (2015) additionally obtains hardness over characteristic 2. However, we have kept the above result in its entirety for the simplicity of its proof.*

For the Moore determinant, by Theorem 7.4, we obtain hardness for all characteristics.

COROLLARY 8.4. *There exists a constant c such that the following holds. For $S = cn^2$, computing the $(1,S)^{th}$ entry of the Moore determinant for $2n \times 2n$ matrices with entries from $M_S(\mathbb{Z})$ is* GapP*-complete w.r.t. polynomial-time Turing reductions. Given a finite field $\mathbb{F}$ of any characteristic $k > 1$, computing the $(1,S)^{th}$ of the Moore determinant for $2n \times 2n$ matrices over $M_S(\mathbb{F})$ is hard w.r.t. polynomial-time Turing reductions for* $\mathrm{Mod}_k\mathrm{P}$.

PROOF (Corollary 8.4). The result follows from Theorem 7.4 and the following observations: computing $H(G)$ over the rationals on an input graph G is GapP-complete w.r.t. polynomial-time Turing reductions; similarly, computing $H(G)$ over a field $\mathbb{F}$ of characteristic k (including $k = 2$) is hard for $\mathrm{Mod}_k\mathrm{P}$ w.r.t. polynomial-time Turing reductions. □

Acknowledgements

The authors would like to thanks the anonymous reviewers of Computational Complexity for their careful reading of the manuscript and their comments. This work was done when the second author was a graduate student at the Institute of Mathematical Sciences. A previous version of this paper appeared in the *ACM Symposium on the Theory of Computing* conference in 2010 (Arvind & Srinivasan 2010).

References

Vikraman Arvind, Pushkar S. Joglekar & Srikanth Srinivasan (2009). 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*, 25–36. URL http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2009.2304.

Vikraman Arvind & Srikanth Srinivasan (2010). 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*, 677–686. URL http://doi.acm.org/10.1145/1806689.1806782.

Helmer Aslaksen (2009). Quaternionic determinants. *The Mathematical Intelligencer* **18**(3), 57–65. ISSN 0343-6993. URL http://dx.doi.org/10.1007/BF03024312.

David A. Mix Barrington (1989). Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. *J. Comput. Syst. Sci.* **38**(1), 150–164. URL http://dx.doi.org/10.1016/0022-0000(89)90037-8.

Alexander Barvinok (2000). New Permanent Estimators via Non-Commutative Determinants. URL http://arxiv.org/abs/math/0007153.

Richard Beigel & John Gill (1992). Counting Classes: Thresholds, Parity, Mods, and Fewness. *Theor. Comput. Sci.* **103**(1), 3–23. URL http://dx.doi.org/10.1016/0304-3975(92)90084-S.

Markus Bläser (2015). Noncommutativity makes determinants hard. *Inf. Comput.* **243**, 133–144. URL http://dx.doi.org/10.1016/j.ic.2014.12.010.

Steve Chien, Prahladh Harsha, Alistair Sinclair & Srikanth Srinivasan (2011). 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*, 499–508. URL http://doi.acm.org/10.1145/1993636.1993703.

Steve Chien, Lars Eilstrup Rasmussen & Alistair Sinclair (2003). Clifford algebras and approximating the permanent. *J. Comput. Syst. Sci.* **67**(2), 263–290. URL http://dx.doi.org/10.1016/S0022-0000(03)00010-2.

Steve Chien & Alistair Sinclair (2007). Algebras with Polynomial Identities and Computing the Determinant. *SIAM J. Comput.* **37**(1), 252–266. URL http://dx.doi.org/10.1137/S0097539705447359.

Stephen A. Fenner, Lance Fortnow & Stuart A. Kurtz (1994). Gap-Definable Counting Classes. *J. Comput. Syst. Sci.* **48**(1), 116–148. URL http://dx.doi.org/10.1016/S0022-0000(05)80024-8.

Craig Gentry (2014). Noncommutative Determinant is Hard: A Simple Proof Using an Extension of Barrington's Theorem. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, 181–187. URL http://dx.doi.org/10.1109/CCC.2014.26.

Chris Godsil & Ivan Gutman (1981). On the matching polynomial of a graph. *Algebraic Methods in Graph Theory I* 241–249.

Pavel Hrubeš, Avi Wigderson & Amir Yehudayoff (2010). Relationless Completeness and Separations. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, 280–290. URL http://doi.ieeecomputersociety.org/10.1109/CCC.2010.34.

Narendra Karmarkar, Richard M. Karp, Richard J. Lipton, László Lovász & Michael Luby (1993). A Monte-Carlo Algorithm for Estimating the Permanent. *SIAM J. Comput.* **22**(2), 284–293. URL http://dx.doi.org/10.1137/0222021.

T.Y. Lam (2005). *Introduction to Quadratic Forms over Fields*. American Mathematical Soc. ISBN 9780821872413.

Cristopher Moore & Alexander Russell (2012). Approximating the Permanent via Nonabelian Determinants. *SIAM J. Comput.* **41**(2), 332–355. URL http://dx.doi.org/10.1137/100806709.

Noam Nisan (1991). 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*, 410–418. URL http://doi.acm.org/10.1145/103418.103462.

Ran Raz & Amir Shpilka (2005). Deterministic polynomial identity testing in non-commutative models. *Computational Complexity* **14**(1), 1–19. URL http://dx.doi.org/10.1007/s00037-005-0188-8.

Leslie G. Valiant (1979). The Complexity of Computing the Permanent. *Theor. Comput. Sci.* **8**, 189–201. URL http://dx.doi.org/10.1016/0304-3975(79)90044-6.

Manuscript received 13 March 2016

V. Arvind
The Institute of Mathematical Sciences (HBNI)
C.I.T. Campus, Chennai, India
arvind@imsc.res.in

Srikanth Srinivasan
Department of Mathematics
IIT Bombay
Mumbai, India
srikanth@math.iitb.ac.in