Almost Settling the Hardness of Noncommutative Determinant[*]

Steve Chien[†] Prahladh Harsha[‡] Alistair Sinclair[§] Srikanth Srinivasan[¶]

ABSTRACT

In this paper, we study the complexity of computing the determinant of a matrix over a noncommutative algebra. In particular, we ask the question: "Over which algebras is the determinant easier to compute than the permanent?" Towards resolving this question, we show the following results for noncommutative determinant computation:

- [Hardness] Computing the determinant of an $n \times n$ matrix whose entries are themselves 2×2 matrices over any field of zero or odd characteristic is as hard as computing the permanent over the field. This extends the recent result of Arvind and Srinivasan, which required the entries to be matrices of dimension linear in n.

- [Easiness] The determinant of an $n \times n$ matrix whose entries are themselves $d \times d$ upper triangular matrices can be computed in poly(n^d) time.

Combining the above with the decomposition theorem for finite dimensional algebras (and in particular exploiting the simple structure of 2×2 matrix algebras), we can extend the above hardness and easiness statements to more general algebras as follows. Let A be a finite dimensional algebra over a finite field of odd characteristic with radical $R(A)$.

- [Hardness] If the quotient $A/R(A)$ is noncommutative, then computing the determinant over the algebra A is as hard as computing the permanent.

- [Easiness] If the quotient $A/R(A)$ is commutative, and furthermore $R(A)$ has nilpotency index d (i.e., d is the smallest integer such that $R(A)^d = 0$), then there exists a poly(n^d)-time algorithm that computes determinants over the algebra A.

In particular, for any constant dimensional algebra A over a finite field of odd characteristic, since the nilpotency index of $R(A)$ is at most a constant, we have the following dichotomy theorem: if $A/R(A)$ is commutative then efficient determinant computation is possible, and otherwise determinant is as hard as permanent.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: Miscellaneous

General Terms

Algorithms, Theory

Keywords

determinant, permanent, noncommutative algebras, computational algebra

1. INTRODUCTION

Given a matrix $M = (m_{i,j})$, the determinant of M, denoted by $\det(M)$, is defined by the polynomial $\det(M) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} m_{i,\sigma(i)}$, while the permanent of M, denoted by $\operatorname{per}(M)$, is defined by the polynomial $\operatorname{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$. Though deceptively similar in their definitions, the determinant and permanent behave very differently with respect to computational complexity. As is well known, the determinant of a matrix over any field can be efficiently computed using Gaussian elimination; in fact, determinant continues to be easy even when the entries come from some commutative algebra, not necessarily a field [18, 4, 7, 14]. On the other hand, computing the permanent of a matrix over the rationals was famously shown by Valiant [19] to be #P-complete, and therefore as hard as counting the

[*] A full version of this paper is available at http://arxiv.org/abs/1101.1169

[†] Google Inc. 1600 Amphitheatre Parkway, Mountain View CA 94043, USA. email: schien@google.com. Work done primarily while the author was at Microsoft Research, Silicon Valley.

[‡] Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai 400005, INDIA. email: prahladh@tifr.res.in. Part of this work was done while the author was at Microsoft Research, Silicon Valley and the MIT Computer Science and Artificial Intelligence Laboratory (supported by NSF Grant CCF-0729011).

[§] Computer Science Division, University of California Berkeley, CA 94720, USA. email: sinclair@cs.berkeley.edu. Supported in part by NSF grants CCF-0635153 and CCF-1016896, and by a UC Chancellor's Professorship.

[¶] Institute for Advanced Study, Einstein Drive, Princeton NJ 08540, USA. email: srikanth@math.ias.edu. Supported by NSF Grants CCF-0832797 and DMS-0835373. Part of this work was done while the author was at Microsoft Research, Silicon Valley.

STOC'11, June 6–8, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0691-1/11/06 ...$10.00.

number of satisfying assignments to a Boolean formula, even when the entries are restricted to 0 or 1. Given this state of affairs, it is natural to ask precisely what it is that makes the permanent hard while the determinant is easy. Understanding this distinction in computational complexity between determinant and permanent is a fundamental problem in theoretical computer science.

Motivated by work of Winograd [20] and Hyafil [11], Nisan pioneered the study of noncommutative lower bounds in his groundbreaking 1991 paper [16]. In one of that paper's more important results, Nisan proves that any algebraic branching program (ABP) that computes the determinant of a matrix $M = \{m_{i,j}\}$ over the noncommutative free algebra $\mathbb{F}\langle x_{1,1}, \ldots, x_{n,n}\rangle$ must have exponential size; this then implies a similar lower bound for arithmetic formulas. This contrasts markedly with the many known efficient algorithms for determinant in commutative settings, which include polynomial-sized ABPs [14].

This problem takes on added significance in light of a connection discovered by Godsil and Gutman [10], and developed by Karmarkar et al. [12], between computing determinants and algorithms for approximating the permanent. The full potential of this approach was demonstrated when Chien et al. [5], expanding on work by Barvinok [3], showed that if one can efficiently compute the determinant of an $n \times n$ matrix M whose entries $m_{i,j}$ are themselves matrices of dimension $O(n^2)$, then there is a fully polynomial randomized approximation scheme for the permanent of a 0-1 matrix[1]; similar results were later proven by Moore and Russell [15]. Thus understanding the complexity of noncommutative determinant is of both algorithmic and complexity-theoretic importance.

Nisan's results are rather limited in that they apply only to the free algebra $\mathbb{F}\langle x_{i,}\rangle$ and not to specific finite dimensional algebras (such as those used to approximate the permanent), and because they do not apply to models of computation beyond ABPs and arithmetic formulas. Addressing the first concern, Chien and Sinclair [6] significantly strengthened Nisan's original lower bounds to apply to a wide range of other algebras by analyzing the polynomial identities of those algebras. In particular, they showed that Nisan's ABP lower bound extends to the $d \times d$ upper-triangular matrix algebra over a field of characteristic 0 for any $d > 1$ (and hence to $M_d(\mathbb{F})$, the full $d \times d$ matrix algebra, as well), the quaternion algebra, and several others.

Addressing the second concern, Arvind and Srinivasan [2] recently showed (conditional) noncommutative determinant lower bounds in much stronger models of computation. They showed that unless there exist small circuits to compute the permanent, there cannot exist small noncommutative circuits for the noncommutative determinant. More devastatingly from the algorithmic point of view, they show that computing $\det(M)$ over matrix algebras of dimension linear in n is at least as hard as (exactly) computing the permanent. This result casts serious doubt on whether the determinant-based approaches to approximating the permanent are computationally feasible.

While these collections of results make substantial progress in our understanding of when determinant can be computed over a noncommutative algebra, they are still incomplete in significant ways. First, we do not know whether Arvind and Srinivasan's results rule out algorithms for determinants over constant-dimensional matrix algebras, which are still of use in approximating the permanent [5]. More expansively, we still do not know the answer to what is perhaps the fundamental philosophical question underlying this line of work:

> *Is there any noncommutative algebra over which one can compute determinants efficiently, or is commutativity a necessary condition for such efficient algorithms?*

1.1 Our results

In this paper, we fill in most of these remaining gaps. Our first main result extends Arvind and Srinivasan's results all the way down to 2×2 matrix algebras (see Theorem 3.5 for a more formal statement).

THEOREM 1.1. **[informal]** *Let $M_2(\mathbb{F})$ be the algebra of 2×2 matrices over a field $\mathbb{F}$ with zero or odd characteristic. Then computing the determinant over $M_2(\mathbb{F})$ is as hard as computing the permanent over $\mathbb{F}$.*

The proof of this theorem works by modifying Valiant's original reduction from #3SAT to permanent. One would not expect to be able to modify the reduction to go from #3SAT to determinant over a field $\mathbb{F}$, as there are known polynomial time algorithms in that setting. However, we show that there is just enough noncommutative behavior in $M_2(\mathbb{F})$ to allow Valiant's reduction (slightly modified) to go through.

Given the central role of matrix algebras in ring theory, this allows us to prove similar results for other large classes of algebras. In particular, consider a finite-dimensional algebra A over a finite field $\mathbb{F}$. This algebra has a *radical* $R(A)$, which happens to be a nilpotent ideal of A (see Section 5 for the exact definition of the radical). Combined with classical results from algebra (in particular the simple structure of the 2×2 matrix algebras) the above theorem can be extended as follows to yield our second main result (see Theorem 5.1 for a more formal statement).

THEOREM 1.2. **[informal]** *If A is a fixed[2] finite dimensional algebra over a finite field of odd characteristic such that the quotient $A/R(A)$ is noncommutative, then computing the determinant over A is as hard as computing the permanent.*

In particular, if the algebra is semisimple (i.e, $R(A) = 0$), then the commutativity of A itself fully characterizes the complexity: if A is commutative, there is an efficient algorithm for computing det over A; otherwise, it is at least as hard as computing the permanent. The class of semisimple algebras includes several well-known examples, such as group algebras.

It may be tempting at this point to conjecture that computing determinant over A for some algebra A is feasible if and only if A is commutative. Perhaps surprisingly, we show that this is not the case: in fact, there do exist noncommutative algebras A over which determinant can be computed in polynomial time. In our third main result, we show that computing the determinant where the matrix entries are $d \times d$ upper triangular matrices for constant d is easy. For reasons that will soon be clear, we will state this result more generally, in the language of radicals.

[1] The results in [5] are stated in the language of *Clifford algebras*. This is an equivalent restatement.

[2] By *fixed*, we mean that the algebra is not part of the input; we fix an algebra A and consider the problem of computing the determinant of a given input matrix over A.

THEOREM 1.3. *Given a finite dimensional algebra A and its radical $R(A)$, let d be the smallest value for which $R(A)^d = 0$ (i.e., any product of d elements of $R(A)$ is 0). If $A/R(A)$ is commutative, there is an algorithm for computing* det *over A in time* $\text{poly}(n^d)$.

While this description of the class of algebras that allow efficient determinant computation may seem somewhat abstruse, it does include several familiar algebras. Perhaps most familiar is the algebra $U_d(\mathbb{F})$ of $d \times d$ upper-triangular matrices, for which $R(U_d(\mathbb{F}))^d = 0$. What the result establishes is that the key to whether determinant is computationally feasible is not commutativity alone. For noncommutative algebras, it is still possible that determinant can be efficiently computed, so long as all of the noncommutative elements belong to a nilpotent ideal and thus, in a sense, have a limited "lifespan."

Taken together, the above theorems yield a pleasing dichotomy for constant dimensional algebras over a finite field of odd characteristic. Given any such algebra A of constant dimension D, either $A/R(A)$ is commutative or not. Furthermore, if $A/R(A)$ is commutative, we have that $R(A)$ is nilpotent with nilpotency index at most D, which is a constant. We thus have the following dichotomy: if $A/R(A)$ is commutative then efficient determinant computation is possible, else determinant is as hard as permanent.

Does this yield a complete characterization of algebras over which the determinant can be efficiently computed? Unfortunately, not quite. In particular, what if the dimension D is non-constant (i.e., the algebra is not fixed but given as part of the input), or if the algebra is over a field of characteristic 0? In these cases, the lower bound of Theorem 1.2 and upper bound of Theorem 1.3 are arguably close, but do not match. A complete characterization remains an intriguing open problem.

Organization of the paper:.

After some preliminaries in Section 2, we first prove lower and upper bounds in two concrete settings: in Section 3 we prove a lower bound for 2×2 matrix algebras, and in Section 4 an upper bound for small-dimensional upper triangular matrix algebras. The results on general algebras are in Section 5, followed by some discussion in Section 6.

2. PRELIMINARIES

In this section we define terms and notation that will be useful later.

An (associative) algebra A over a field $\mathbb{F}$ is a vector space over $\mathbb{F}$ with a bilinear, associative multiplication operator that distributes over addition. That is, we have a map $\cdot : A \times A \to A$ that satisfies: (a) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ for any $x, y, z \in A$; (b) $\lambda(x \cdot y) = (\lambda x) \cdot y = x \cdot (\lambda y)$, for any $\lambda \in \mathbb{F}$ and $x, y \in A$; and (c) $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(y + z) \cdot x = y \cdot x + z \cdot x$ for any $x, y, z \in A$. We will assume that all our algebras are *unital*, i.e., they contain an identity element. We will denote this element as 1. For more background on algebras, see Curtis and Reiner's book [8]. A wide range of familiar mathematical objects are algebras; in this paper we will be concerned mainly with the algebra of $d \times d$ matrices over $\mathbb{F}$, which we will denote $M_d(\mathbb{F})$, as well as the algebra of $d \times d$ upper-triangular matrices over $\mathbb{F}$, or $U_d(\mathbb{F})$. Other prominent examples are the free algebra $\mathbb{F}\langle x_i \rangle$, the algebra of polynomials $\mathbb{F}[x_i]$, group algebras over a field, or a field considered as an algebra over itself.

Given an $n \times n$ matrix $M = (m_{i,j})$ whose elements belong to an algebra A, the *determinant* of M, or $\det(M)$, is defined as the polynomial $\det(M) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} m_{i,\sigma(i)}$. Note that when A is noncommutative, the order of the multiplication becomes important. When the order is by row, as above, the resulting object is known as the "Cayley determinant." The permanent of the same matrix is $\text{per}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$. We will denote by $\det_A$ (and per_A) the determinant (and permanent) over an algebra A (i.e., the matrix entries are elements of A).

We recall also the familiar recasting of the determinant and permanent in terms of cycle covers on a graph. Suppose $M = (m_{i,j})$ is an $n \times n$ matrix over an algebra A. Let $G(M)$ denote the weighted directed graph on vertices $1, \ldots, n$ that has M as its adjacency matrix. A permutation $\pi : [n] \to [n]$ from the rows to the columns of M can be identified with the set of edges $(i, \pi(i))$ in the graph $G(M)$; it is easily observed that these edges form a (directed) cycle cover of $G(M)$. Letting $\mathcal{C}(G)$ denote the collection of all cycle covers of $G(M)$, we can write

$$\det(M) = \sum_{C \in \mathcal{C}(G(M))} \text{sgn}(C) m_{1,C(1)} m_{2,C(2)} \cdots m_{n,C(n)} \quad (2.1)$$

$$\text{per}(M) = \sum_{C \in \mathcal{C}(G(M))} m_{1,C(1)} m_{2,C(2)} \cdots m_{n,C(n)},$$

where for a given cycle cover C, $C(i)$ represents the successor of vertex i in C, and $\text{sgn}(C)$ is the sign of C. It is known that $\text{sgn}(C) = (-1)^{n-c}$, with c being the number of cycles in C, and that this is also the sign of the corresponding permutation. We will denote the weight of an edge $e = (x, y)$ as $w(e)$ or $w(x, y)$. Further, for a subset of edges $B = \{(x_1, y_1), \ldots, (x_{|B|}, y_{|B|})\}$ of a cycle cover C with $x_i < x_{i+1}$, we can define the weight of B as $w(B) = \prod_{i=1}^{|B|} w(x_i, y_i)$. (Note that the product is in order by source vertex.) Thus $w(C) = \prod_i m_{i,C(i)}$ is the weight of the cycle cover, and the product $\text{sgn}(C) \prod_i m_{i,C(i)}$ is the *signed weight* of C.

We briefly recall the definitions of some complexity classes (the Complexity Zoo[3] is an excellent reference). A function $f : \{0,1\}^n \to \mathbb{N}$ is said to be in #P if there exists a polynomial time nondeterministic Turing Machine M such that $f(x)$ equals the number of accepting paths of M on input x. Given a fixed number $m \in \mathbb{N}$, a function $f : \{0,1\}^n \to \{0,1\}$ is said to be in $\text{Mod}_m P$ if there exists a polynomial time nondeterministic Turing Machine M such that $f(x) = 1$ iff the number of accepting paths of M on input x is not divisible by m. It is well known that #3SAT, the problem of computing the number of satisfying assignments of a given 3-SAT formula φ, is hard for #P, and the problem of counting the number of satisfying assignments of φ modulo m is hard for $\text{Mod}_m P$.

3. THE LOWER BOUND FOR 2 X 2 MATRIX ALGEBRAS

In this section, we show our lower bound for 2×2 matrix algebras. Our proof is based on Valiant's seminal reduction from #3SAT to permanent, as modified by Papadimitriou [17] and also described in the textbook of Arora and

[3] http://qwiki.stanford.edu/index.php/Zooref

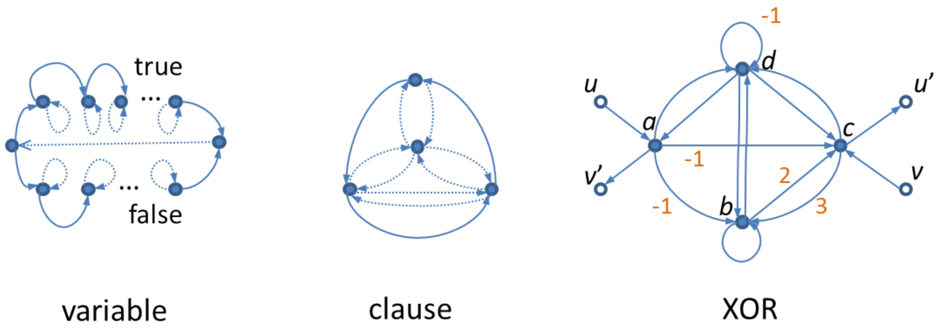

Figure 1: Gadgets used in proving Valiant's lower bound. All edges have weight 1 unless noted otherwise. For the variable and clause gadgets, the solid (not dotted) edges are called *(vertex) external edges* or *(clause) external edges*. Note that the number of external edges in a variable gadget is not fixed, and need not be the same for the True and False halves of the gadget.

Barak [1]. We first give a self-contained description of this reduction, before detailing our modifications of it.

3.1 Valiant's lower bound for the permanent

Valiant's reduction is from #3SAT to permanent; given a #3SAT formula φ on n variables and m clauses, he constructs a weighted directed graph G_φ on poly(n, m) vertices such that the number of satisfying assignments of φ is equal to constant × per$(M(G_\varphi))$, where $M(G_\varphi)$ is the adjacency matrix of G_φ. The key components of G_φ are the variable, clause, and XOR gadgets shown in Figure 1.[4] The idea is that there will be a relation between satisfying assignments of φ and cycle covers of G_φ; moreover, for each satisfying assignment, the total weight of its corresponding cycle covers will be the same.

Before defining G_φ itself, we first work with a preliminary graph G_φ^0 that contains n variable gadgets and m clause gadgets, but no XOR gadgets; all of the gadgets are disjoint from each other. For the moment, the number of external edges in each of the variable gadgets is unimportant. In analyzing G_φ^0, we will use the following:

Lemma 3.1. *The following hold for the gadgets in Figure 1: (a) A variable gadget has exactly two cycle covers. Each cycle cover contains one long cycle using all of the external edges on one side of the gadget and the long middle edge, as well as all the self-loops on the other side of the gadget. (b) In a clause gadget, there is no cycle cover that uses all three external edges. For every* proper *subset S of the external edges in a clause gadget, there is exactly one cycle cover that contains exactly the edges in S; this cycle cover has weight* 1.

As all $n + m$ gadgets in G_φ^0 are disjoint, any cycle cover of G_φ^0 will be a union of $n + m$ smaller cycle covers—namely, one for each gadget. The choice of cycle cover for each gadget defines the value of each variable and which literals are satisfied in each clause.

More precisely, for a variable gadget, let the term *True cycle cover* denote the cycle cover containing the external edges on the True side of the gadget. Analogously, the *False cycle cover* refers to the cycle cover containing the external edges on the False side of the gadget. The idea is that a cycle cover of G_φ^0 sets a variable to T or F by choosing either the True or False cycle cover. Meanwhile, for clause gadgets, the intention is that each external edge will correspond to one of the three literals in the clause, and an external edge is used in a cycle cover if and only if the corresponding literal is set to F (i.e. the corresponding literal is *not* satisfied). Since no cycle cover can contain all three external edges of a clause gadget, in this interpretation at least one of the literals in the clause must be satisfied.

We say a cycle cover C of G_φ^0 is *consistent* if (1) whenever C contains the True cycle cover of the gadget for a variable x_k, it contains *all* clause external edges for instances of the negative literal $\overline{x_k}$ and *no* clause external edges for instances of the positive literal x_k, and (2) conversely, whenever C contains the False cycle cover for x_k, it contains all clause external edges for instances of x_k but no clause external edges for instances of $\overline{x_k}$. A consistent cycle cover therefore does not "cheat" by claiming to set x_k to T (for example) in a variable gadget but to F in a clause gadget. This is close to what we want:

Lemma 3.2. *The number of satisfying assignments of φ is equal to the total weight of consistent cycle covers of G_φ^0.*

Proof. This follows by combining the natural bijection between satisfying assignments and consistent cycle covers and the fact from Lemma 3.1 that every cycle cover of a clause gadget has weight 1. □

Of course, nothing about G_φ^0 guarantees that a cycle cover must be consistent, and in fact many inconsistent covers exist. To fix this, we make crucial use of the XOR gadgets to obtain the final graph G_φ.

The graph G_φ is constructed as shown in Figure 2 (left). It has the same n variable gadgets and m clause gadgets as G_φ^0, with the gadget for each variable x_k having as many True external edges as there are instances of x_k in φ, and as many False external edges as there are instances of $\overline{x_k}$. Now, however, for each appearance of a literal x_k or $\overline{x_k}$ in a given clause, an XOR gadget is used to replace the corresponding external edge in that clause gadget and a distinct external

[4] We follow a convention from [1] in allowing gadgets to sometimes have multiple edges between the same two vertices. While technically prohibited in a graph defined by a matrix, this can be fixed by adding an extra node in these edges.

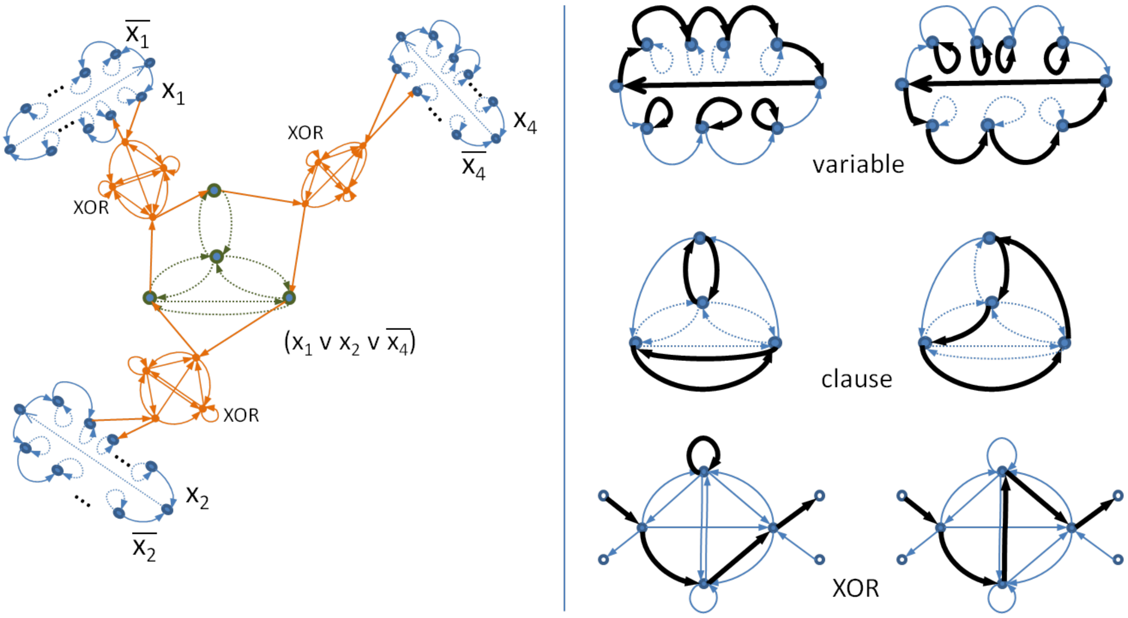

Figure 2: Left: Subgraph of G_φ corresponding to clause $(x_1 \vee x_2 \vee \overline{x_4})$, with the clause gadget in the center. Three variable gadgets are connected to the clause gadget via XOR gadgets. Right: Examples of how gadgets may have cycle covers of different sign.

edge on the appropriate side of the variable gadget for x_k. The role of the XOR gadgets is to neutralize the inconsistent cycle covers of G^0_φ while still maintaining the property that each satisfying assignment of φ contributes the same to the total weight of cycle covers. This leads to the description of the final graph G_φ itself.

We now state the important properties of the XOR gadget, the key component of Valiant's proof.

Lemma 3.3. *Suppose a graph G contains edges (u, u') and (v, v'), with all four vertices distinct. Suppose now that the edges (u, u') and (v, v') are replaced by an XOR gadget as shown in Figure 1, resulting in a new graph G' (with four new vertices a, b, c and d). Let $\mathcal{C}_{u\setminus v}$ be the set of cycle covers containing (u, u') but not (v, v'), and $w_{u\setminus v} = \sum_{C\in\mathcal{C}_{u\setminus v}} w(C)$ be their total weight. Let $\mathcal{C}_{v\setminus u}$ and $w_{v\setminus u} = \sum_{C\in\mathcal{C}_{v\setminus u}} w(C)$ be defined analogously. Then there exist two disjoint sets of cycle covers of G' with total weight $4w_{u\setminus v}$ and $4w_{v\setminus u}$, while all cycle covers of G' not in these sets have total weight 0.*

The proof is omitted, as we will state and prove our own modified version in Section 3.2 (see Lemma 3.8).

This leads to the following:

Theorem 3.4. **[Valiant]** *Given a 3-SAT formula φ and the graph G_φ as described, $\mathrm{per}(G_\varphi) = 4^{3m}S$, where S is the number of satisfying assignments of φ.*

We omit the formal proof, but give some of the intuition. Beginning with G^0_φ, we begin adding XOR gadgets one at a time. When a pair of edges is replaced by an XOR gadget, any cycle covers that are consistent with respect to that pair of edges are turned into a set of cycle covers whose total weight is four times the original weight. All other cycle covers in the new graph have total weight 0. This continues until each of the $3m$ XOR gadgets are added, at which point the original consistent cycle covers have become a set of cycle covers with total weight 4^{3m} while all other cycle covers in the final graph have weight 0. The total weight of the cycle covers in the final graph is therefore $4^{3m}S$, as required.

3.2 Our construction

In this section we prove the following formal version of Theorem 1.1 stated in the introduction.

Theorem 3.5. *Let $\mathbb{F}$ be a field of characteristic $p \neq 2$. If $p = 0$, computing $\det_{M_2(\mathbb{F})}$ is #P-hard. On the other hand, if $p > 2$, computing $\det_{M_2(\mathbb{F})}$ is $\mathrm{Mod}_p P$-hard.*

Our proof is also a reduction from #3SAT (or Mod_p-SAT in the case of positive odd characteristic) and is based on Valiant's framework as described in the previous subsection. Given a 3SAT formula φ, we wish to construct a directed graph H_φ with weights belonging to $M_2(\mathbb{F})$ such that the number of satisfying assignments of φ can be computed from $\det(M(H_\varphi))$, as expressed in equation (2.1) above. We will first describe the graph and then prove its correctness.

A very naive but instructive first try would be to simply use the graph G_φ from Valiant's construction, replacing each edge weight $w \in \mathbb{F}$ with wI_2, where I_2 is the 2×2 identity matrix. This fails, of course, because of the factor $\mathrm{sgn}(C)$ inside the summation, which is based on the parity of the number of cycles in C. The immediate problem is that each of the three types of gadgets could conceivably use an odd or even number of cycles. As shown in Figure 2 (right), variable gadgets may have a different number of self-loops on different sides; clause gadgets may use one or two cycles depending on which external edges are chosen; and XOR gadgets show similar behavior.

Fortunately, these problems can be overcome if we also allow ourselves to modify the edge weights, and crucially, use the noncommutative structure available in $M_2(\mathbb{F})$. This

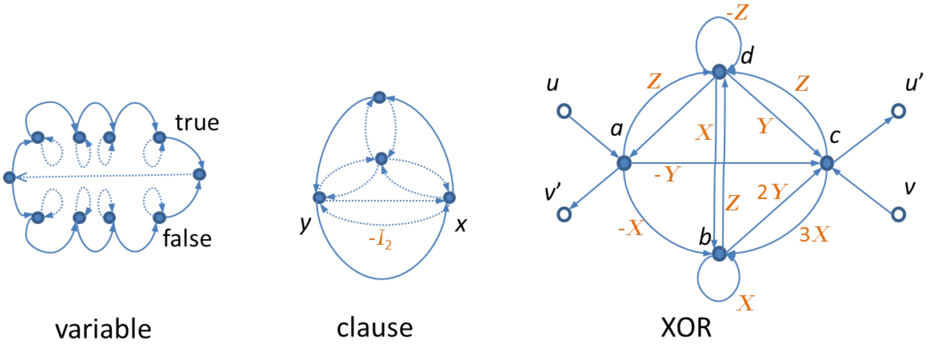

Figure 3: Modified gadgets.

results in the gadgets shown in Figure 3. We now define two graphs, a preliminary graph H^0_φ and final graph H_φ, in analogy with G^0_φ and G_φ from Section 3.1. The new graphs H^0_φ and H_φ will be constructed in the same manner as G_φ, only using the modified gadgets from Figure 3 instead of the original gadgets in Figure 1.

The rough idea behind these gadgets is that with the new weights, each resulting cycle cover of a gadget of the "wrong" sign will have an extra -1 sign from its edge weights. The determinant is then essentially the same as the permanent. We now explain the changes in more detail.

For variable gadgets, the fix is easy: all we have to do is make sure that both sides of the gadget have (say) an even number of vertices, and hence an even number of self loops. This can be accomplished by adding, if necessary, a new vertex and appropriate new edges on one or both sides. The new external edges, if any, will not be connected to any of the clause gadgets.

For clause gadgets, we need to address the problem that some cycle covers have only one cycle, while others have two. Here we benefit from the observation that one of the edges, (x, y) in Figure 3, is used only in cycle covers with two cycles. Thus we can correct for parity by changing the sign of this edge from I_2 to $-I_2$; as a result, every cycle cover of a clause gadget has the same signed weight.

For XOR gadgets, simply changing the edge weights to scalar multiples of I_2 is insufficient. (Indeed, Valiant presciently noticed this in 1979!) However, we can rescue the construction by using more sophisticated matrix-valued edge weights instead. In particular, we define the following three 2×2 matrices:

$$X = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}; \quad Y = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}; \quad Z = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{3.1}$$

We then modify the weights of the edges between vertices a, b, c and d. Specifically, each edge entering vertex b has its weight multiplied by X; each edge entering c has its weight multiplied by Y, and each edge entering d has its weight multiplied by Z.

Now, with H_φ defined, we prove that computing $\det(H_\varphi)$ is equivalent to computing the number of satisfying assignments of φ. We first observe the following analogue of Lemma 3.2.

Lemma 3.6. *Let $\mathcal{C}_{\text{con}}$ be the set of all consistent cycle covers of H^0_φ. Then there exists $z \in \{1, -1\}$ such that for all $C \in \mathcal{C}_{\text{con}}$, we have $\operatorname{sgn}(C)w(C) = zI_2$.*

Proof. As in the proof of Lemma 3.2, there is a bijection between satisfying assignments of φ and consistent cycle covers of H^0_φ. We need to show that each of these cycle covers has the same signed weight. For such a cycle cover $C \in \mathcal{C}_{\text{con}}$ we have $\operatorname{sgn}(C) = (-1)^{n^0_H - c(C)}$, where n^0_H is the number of vertices in H^0_φ and $c(C)$ is the number of cycles in C. We further know that $(-1)^{c(C)} = (-1)^{p+m+q}$, where p is the number of cycles used to cover the n variable gadgets, m is the number of clauses, and q is the number of times C uses two cycles to cover a clause gadget. Since we assumed p to be even, we have $\operatorname{sgn}(C) = (-1)^{n^0_H + m + q}$.

On the other hand, $w(C)$ is the product of the edge weights of C. All of these weights are I_2 except for the $w(x, y)$ in the clause gadget, which has weight $-I_2$ and shows up when C uses two edges for a clause gadget. Thus $w(C) = (-1)^q I_2$, and $\operatorname{sgn}(C)w(C) = (-1)^{n^0_H + m} I_2$, which is independent of the cycle cover C. (Hence, $\sum_{C \in \mathcal{C}_{\text{con}}} \operatorname{sgn}(C)w(C) = (-1)^{n^0_H + m} S I_2$, where S is the number of satisfying assignments of φ.) □

Without loss of generality, we can assume from here on that the sign z is positive, as we can insert a new vertex within an edge so that $n^0_H + m$ is even.

We now prove the following useful identities for XOR gadgets, which can be verified by hand.

Lemma 3.7. *Let M be the adjacency matrix for the XOR gadget, i.e.,*

$$M = \begin{pmatrix} 0 & -X & -Y & Z \\ 0 & X & 2Y & Z \\ 0 & 3X & 0 & Z \\ I_2 & X & Y & -Z \end{pmatrix}.$$

Letting $M_{I,J}$ indicate the minor of M with rows in set I and columns in set J removed, we have (1) $\det(M_{3,1}) = -4I_2$; (2) $\det(M_{1,3}) = -4J_2$; (3) $\det(M) = \det(M_{1,1}) = \det(M_{3,3}) = \det(M_{\{1,3\},\{1,3\}}) = 0$, where $J_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Now consider a graph G with vertices labeled $1, \ldots, n_G$ and weights in $M_2(\mathbb{F})$. Suppose G contains vertex-disjoint edges (u, u') and (v, v'), each with weight I_2. Suppose now that the edges (u, u') and (v, v') are replaced by an XOR gadget as shown in Figure 3. This results in a new graph G', with four new vertices a, b, c and d, which we number $n_G + 1, \ldots, n_G + 4$. We now define a mapping ψ from $\mathcal{C}(G)$ to subsets of $\mathcal{C}'(G)$ as follows: given cycle covers $C \in \mathcal{C}(G)$ and $C' \in \mathcal{C}(G')$, then $C' \in \psi(C)$ if and only if (1) for all edges $e \in C\setminus\{(u, u'), (v, v')\}$, we have $e \in C'$; (2) $(u, u') \in C$ if and only if $(u, a), (c, u') \in C'$; and (3) $(v, v') \in C$ if and only if $(v, c), (a, v') \in C'$.

This leads to the following analogue of Lemma 3.3:

Lemma 3.8. *Let $\mathcal{C}_{u\setminus v} = \{C \in \mathcal{C}(G) : (u, u') \in C, (v, v') \notin C\}$ be the set of cycle covers of G containing (u, u') but not (v, v'), and $\mathcal{C}_{v\setminus u} = \{C \in \mathcal{C}(G) : (v, v') \in C, (u, u') \notin C\}$. Then there exists a mapping ψ from $\mathcal{C}(G)$ to subsets of $\mathcal{C}'(G)$ such that $\psi(C_1) \cap \psi(C_2) = \emptyset$ for all $C_1, C_2 \in \mathcal{C}(G)$ and (1) for any $C \in \mathcal{C}_{u\setminus v}$, the total weight of $\psi(C)$ is $\sum_{C'\in\psi(C)} \mathrm{sgn}(C')w(C') = 4\mathrm{sgn}(C)w(C)$, (2) for any $C \in \mathcal{C}_{v\setminus u}$, $\sum_{C'\in\psi(C)} \mathrm{sgn}(C)w(C') = 4\mathrm{sgn}(C)w(C)J_2$, and (3) the remaining cycle covers in G' have total weight $\sum_{C'\notin\psi(C)\forall C\in\mathcal{C}_{u\setminus v}\cup\mathcal{C}_{v\setminus u}} \mathrm{sgn}(C')w(C') = 0$.*

Proof. We start by proving (1). Fix any $C \in \mathcal{C}_{u\setminus v}$. Notice that $\psi(C)$ consists of all $C' \in \mathcal{C}(G')$ that contain (u, a), (c, u') and all of C's edges except (u, u'). Call this set of common edges E_C; by the assumption that $w(u, u') = I_2$, $w(E_C) = w(C)$. The set $\psi(C)$ consists of all possible ways of completing E_C to a cycle cover C' of G' by adding edges to G' so that every vertex has indegree and outdegree 1. Within E_C, the only vertices with deficient degree are a, b, c and d. Vertices b and d have indegree and outdegree 0, while a has indegree 1 and outdegree 0, and c has indegree 0 and outdegree 1. Note that the edges (u, a) and (c, u') must belong to the same cycle in C', and so the edges in E_C form zero or more completed cycles and an incomplete cycle from c to a. The number of completed cycles is $c(C) - 1$, where $c(C)$ is the number of cycles in C.

We thus need to add three edges matching the vertices $\{a, b, d\}$ to the vertices $\{b, c, d\}$; call these three edges E_{XOR}, so that $E_C \cup E_{\text{XOR}}$ forms a cycle cover C'. The weight of C' is therefore $w(C') = w(E_C)w(E_{\text{XOR}})$. The sign of C' is $(-1)^{n+4-c(C')}$, where $c(C')$ is the number of cycles in C'. We can see that $c(C')$ is the sum of the number of completed cycles in E_C and the number of cycles among $\{a, b, c, d\}$ assuming the existence of an edge from c to a. Hence $c(C') = c(C)-1+c(E_{\text{XOR}}\cup\{(c, a)\})$, and so $\mathrm{sgn}(C') = -\mathrm{sgn}(C)(-1)^{4-c(E_{\text{XOR}}\cup\{(c,a)\})} = -\mathrm{sgn}(C)\mathrm{sgn}(E_{\text{XOR}} \cup \{(c, a)\})$. Thus, $\sum_{C'\in\psi(C)} \mathrm{sgn}(C')w(C') = - \mathrm{sgn}(C)w(E_C) \sum_{C'\in\psi(C)} \mathrm{sgn}(E_{\text{XOR}} \cup \{(c, a)\})w(E_{\text{XOR}}) = -\mathrm{sgn}(C)w(E_C) \det(M_{3,1})$. From Lemma 3.7, this is $4\mathrm{sgn}(C)w(E_C) = 4\mathrm{sgn}(C)w(C)$, as required.

The proof of (2) proceeds similarly, except that $\psi(C)$ contains (v, c) and (a, v') instead of (u, a) and (c, v'). The set of common edges then has an incomplete path from a to c. As a result, we end up with $\sum_{C'\in\psi(C)} \mathrm{sgn}(C')w(C') = -\mathrm{sgn}(C)w(E_C) \det(M_{1,3}) = 4\mathrm{sgn}(C)w(C)J_2$.

To prove (3), we observe that a cycle cover in $\mathcal{C}(G')$ that contains (u, a) and (c, u') but not (v, c) or (a, v') must fall into $\psi(C)$ for some $C \in \mathcal{C}_{u\setminus v}$; similarly, any cycle cover containing (v, c) and (a, v') but not (u, a) or $c, u')$ must fall into $\psi(C)$ for some $C \in \mathcal{C}_{v\setminus u}$. These were already accounted for in the proofs of (1) and (2), so we can concentrate only on the leftover cycle covers. Partition these leftover cycle covers into equivalence classes based on their edge sets excluding edges wholly within $\{a, b, c, d\}$; namely $C'_1 \sim C'_2$ if and only if $C'_1 \setminus \{a, b, c, d\} \times \{a, b, c, d\} = C'_2 \setminus \{a, b, c, d\} \times \{a, b, c, d\}$. For any equivalence class, its cycle covers must either all (a) contain none of these four edges, (b) contain (u, a) and (a, v') only, (c) contain (v, c) and (a, v') only, or (d) contain all four edges.

Up to sign, the total weights of those equivalence classes in (a) contain a factor of $\det(M)$, those in (b) contain a factor $\det(M_{1,1})$, those in (c) contain a factor $\det(M_{3,3})$, and those in (d) contain a factor $\det(M_{\{1,3\},\{1,3\}})$. From Lemma 3.7, all four of these determinants are 0, and so the total weight of the cycle covers in any equivalence class is 0, as is therefore the total weight of all the leftover cycle covers. □

With this in hand, we can prove the following key result:

Theorem 3.9. *Given a 3SAT formula φ with S satisfying assignments, let the graph H_φ with weights in $M_2(\mathbb{F})$ be as defined above. Then $\det(H_\varphi) = aI_2 + bJ_2$, where $a + b = 4^{3m}S$.*

Proof. The structure of the proof is similar to the sketch given after Theorem 3.4, though with extra care needed to handle the complications of working with matrices. In the end, each cycle cover of G ends up with weight $4^{3m}I_2$ or $4^{3m}J_2$, giving the result.

Let us start with H^0_φ, which we know from Lemma 3.6 has $\det(H^0_\varphi) = SI_2$. In particular, for each satisfying assignment of φ, there is a consistent cycle cover of H^0_φ of weight I_2. There exist $3m$ pairs of edges in H^0_φ that when replaced by XOR gadgets will convert H^0_φ to H_φ; each of these pairs contains an external edge in a clause gadget and an external edge in a variable gadget referring to the same literal.

Consider what happens when we replace one of the above-mentioned edge pairs with an XOR gadget, forming a new graph H^1_φ. From Lemma 3.8, each cycle cover C that is consistent on this edge pair in H^0_φ will be mapped to $\psi(C)$, a set of cycle covers in the new graph whose total signed weight will either be $4I_2$ or $4J_2$. Further, since all of these sets $\psi(C)$ are disjoint and all other cycle covers have total signed weight 0, the total signed weight of all cycle covers in H^1_φ is $\sum_{C\in\mathcal{C}^1_{\text{con}}(G)} 4K_2(C)$, where $\mathcal{C}^1_{\text{con}}(G)$ are those cycle covers of G that are consistent on this edge pair, and $K_2(C)$ is either I_2 or J_2.

Now suppose a second edge pair is replaced with an XOR gadget, resulting in the graph H^2_φ. Consider a cycle cover C of H^0_φ in $\psi(C)$ that is consistent on both the first and second edge pairs. Then each cycle cover of H^1_φ in $\psi(C)$ will be mapped to a set of cycle covers $\psi(\psi(C))$ of H^2_φ, with signed weight that is $4I_2$ or $4J_2$ multiple of its signed weight in H^1_φ. The set $\psi(\psi(C))$ therefore has total signed weight of either $16I_2$ or $16J_2$, since all of the images of ψ are disjoint. Once again, the total signed weight of all cycle covers in H^2_φ is $\sum_{C\in\mathcal{C}^{1,2}_{\text{con}}(G)} 16K_2(C)$, where $\mathcal{C}^{1,2}_{\text{con}}(G)$ is the set of cycle covers of G consistent on both edge pairs.

Carrying this out over all $3m$ edge pairs to reach H_φ, we see that every consistent cycle cover of H^0_φ becomes a disjoint set of cycle covers in H_φ of total signed weight $4^{3m}I_2$ or $4^{3m}J_2$, while all other cycle covers in H_φ have total weight 0. The total weight over all original consistent cycle covers is $\sum_{C\in\mathcal{C}_{\text{con}}(G)} 4^{3m}K_2(C)$. This therefore takes the form given in the theorem. □

This completes the proof of Theorem 3.5.

4. DETERMINANT COMPUTATION OVER UPPER TRIANGULAR MATRIX ALGEBRAS

In this section, we consider the problem of computing the determinant over the algebra of upper triangular matrices of dimension d. We show that the determinant over these algebras can be computed in time $N^{O(d)}$, where N denotes the size of the input. We will later generalize this result to arbitrary algebras to yield Theorem 5.4.

Given a field $\mathbb{F}$, recall that $U_d(\mathbb{F})$ denotes the algebra of upper triangular matrices of dimension d with entries from $\mathbb{F}$.

THEOREM 4.1. *Let $\mathbb{F}$ be a field. There exists a deterministic algorithm that, when given as input an $n \times n$ matrix M with entries from $U_d(\mathbb{F})$, computes the determinant of M in time* $\mathrm{poly}(N^d)$*, where N is the size of the input.*

PROOF. The algorithm is simple. We write out the expression for the determinant of M and note that each entry of $\det(M)$ may be written as the sum of $n^{O(d)}$ many determinants of matrices with entries from the underlying field. Since each of these can be computed in time $N^{O(1)}$, we obtain an $N^{O(d)}$-time algorithm for our problem.

Let $M = (m_{i,j})$, where $m_{i,j} \in U_d(\mathbb{F})$ for each $i, j \in [n]$. Given $m \in U_d(\mathbb{F})$, we use $m(p, q)$ to denote the (p, q)th entry of m. We have

$$\det(M) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) m_{1,\sigma(1)} m_{2,\sigma(2)} \cdots m_{n,\sigma(n)}$$

Consider a product of matrices $m = m_1 \cdots m_n$ where each $m_i \in U_d(\mathbb{F})$. For $p, q \in [d]$ such that $p \leq q$, we may write the (p, q)th entry of m as

$$\begin{aligned} m(p,q) &= \sum_{k_1,k_2,\ldots,k_{n-1} \in [d]} m_1(p,k_1) m_2(k_1,k_2) \cdots m_n(k_{n-1},q) \\ &= \sum_{p \leq k_1 \leq \cdots \leq k_{n-1} \leq q} m_1(p,k_1) m_2(k_1,k_2) \cdots m_n(k_{n-1},q) \end{aligned} \tag{4.1}$$

where the last equality follows since $m_i(k, l) = 0$ unless $k \leq l$. Note that the number of terms in the summation in (4.1) is equal to the number of increasing sequences of length n consisting of elements from $[d]$ and is bounded by $n^{O(d)}$.

Fix any $p, q \in [d]$ such that $p \leq q$. By (4.1), we may write $\det(M)(p, q)$ as

$$\sum_{p \leq k_1 \leq \cdots \leq k_{n-1} \leq q} \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \cdot m_{1,\sigma(1)}(p,k_1) \cdot m_{2,\sigma(2)}(k_1,k_2) \cdots m_{n,\sigma(n)}(k_{n-1},q).$$

We now note that each of the inner summations may be written as the determinant of an appropriate matrix over the underlying field. Fix any $\overline{k} = (k_1, \ldots, k_{n-1})$ satisfying $p \leq k_1 \leq k_2 \leq \cdots \leq k_{n-1} \leq q$. Denote by $M_{\overline{k}}$ the matrix $(m_{i,j}(k_{i-1}, k_i))_{i,j}$, where k_0 denotes p and k_n denotes q. It follows from (4.2) that $\det(M)(p, q) = \sum_{\overline{k}} \det(M_{\overline{k}})$.

Note that the matrices $M_{\overline{k}}$ are $n \times n$ matrices with entries from the underlying field, and hence their determinants can be computed in time $N^{O(1)}$. Therefore, we can compute $\det(M)(p, q)$, for each p, q, in time $n^{O(d)} \cdot N^{O(1)} = N^{O(d)}$. The result follows. □

5. DETERMINANT COMPUTATION OVER GENERAL ALGEBRAS

We now consider the problem of computing the determinant of an $n \times n$ matrix with entries from a general finite-dimensional algebra A of dimension D over a field $\mathbb{F}$ that is either finite or the rationals. We consider the algorithmic question of computing the determinant over A, where A is a *fixed* algebra (and hence of constant dimension) such as $M_2(\mathbb{F})$.

We prove a strong dichotomy for finite fields of characteristic $p > 2$. For any fixed algebra A, we show, based on the structure of the algebra, that either the determinant over A is polynomial time computable, or computing the determinant over A is $\mathrm{Mod}_p P$-hard. For the rationals, we do not obtain a dichotomy but our results yield upper and lower bounds for many interesting classes of algebras.

We first recall a few basic facts about the structure of finite dimensional algebras. An algebra is *simple* if it is isomorphic to a matrix algebra (possibly of dimension 1) over a field extension of $\mathbb{F}$. An algebra is *semisimple* if it can be written as the direct sum of simple algebras.[5]

Recall that a *left ideal* in an algebra A is a subalgebra I of A such that for any $x \in I$ and $a \in A$, we have $ax \in I$; a *right ideal* is defined similarly. An ideal I is said to be *nilpotent* if there exists an $m \geq 1$ such that the product of any m elements from I is 0. The *radical of* A, denoted $R(A)$, is defined to be the ideal generated by all the nilpotent left ideals of A. We list some well-known properties of the radical (see [8, Chapter IV]): (a) the radical is a left and right ideal in A; (b) the radical is nilpotent; and (c) $A/R(A)$ is semisimple. The least d such that the product of any d elements of $R(A)$ is 0 is called the *nilpotency index* of $R(A)$.

An algebra A is a *semidirect sum* of subalgebras B_1 and B_2 if $A = B_1 \oplus B_2$ as a vector space; we denote this as $A = B_1 \oplus' B_2$. The *Wedderburn-Malcev theorem* tells us that any algebra is a semidirect sum of its radical with a subalgebra. We refer to such a decomposition as a *Wedderburn-Malcev decomposition.*

We start with the hardness result, which is a formal version of Theorem 1.2 in the introduction.

THEOREM 5.1. *Let A denote any fixed algebra over a finite field $\mathbb{F}$ of characteristic $p > 2$. If $A/R(A)$ is noncommutative, computing the determinant over A is* $\mathrm{Mod}_p P$*-hard.*

PROOF. Consider the problem of computing the determinant over an algebra A such that $A/R(A)$, the "semisimple part" of A, is noncommutative. Since $A/R(A)$ is semisimple, we know that $A/R(A) \cong \bigoplus_i A_i$, where each A_i is a simple algebra, and hence isomorphic to a matrix algebra over a field extension of $\mathbb{F}$. If each of the A_i's is a matrix algebra of dimension 1 (that is, each A_i is simply a field extension of $\mathbb{F}$), then $A/R(A)$ is commutative. Hence, w.l.o.g., we assume that A_1 has dimension greater than 1. Moreover, by the Wedderburn-Malcev theorem we know that A contains a subalgebra $B \cong A/R(A)$. Thus, the algebra A_1 is isomorphic to a subalgebra of A. Thus, Theorem 3.5 immediately implies that computing the determinant over A is $\mathrm{Mod}_p P$-hard. □

[5] This is not the standard definition of semisimplicity in the case of infinite fields. However, we will only use it in the case that $\mathbb{F}$ is finite (cf., [13].)

5.1 The upper bound

In this section, we show that if $A/R(A)$ is commutative then the determinant over A is efficiently computable. However, we present our result in somewhat greater generality. We assume that the algebra A is presented to the algorithm along with the input as follows: we are given a (vector space) basis $\{a_1, \dots, a_D\}$ for A along with the pairwise products $a_i a_j$ for every $i, j \in [D]$. Let d denote the nilpotency index of $R(A)$.

The Wedderburn-Malcev theorem tells us that the algebra $A = B \oplus' R(A)$, where B is a semisimple subalgebra of A isomorphic to $A/R(A)$, and hence commutative.

We use without explicit mention the following result, which was stated in the work of Chien and Sinclair [6] (and implicit in that of Mahajan and Vinay [14] and several other works): There is a deterministic algorithm which, when given any commutative algebra A of dimension D and an $n \times n$ matrix over A as input, computes the determinant of A in time $\mathrm{poly}(n, D)$.

We start with two simple lemmas, the proofs of which are deferred to the full version.

Lemma 5.2. *There is a deterministic polynomial time algorithm which, when given an algebra A as input, computes the nilpotency index of A.*

Lemma 5.3. *Let A be a finite-dimensional algebra with Wedderburn-Malcev decomposition $A = B \oplus' R(A)$. Then $1 \in B$.*

These lemmas and a generalization of Theorem 4.1 yield the following:

Theorem 5.4. *There exists a deterministic algorithm that, when given as input an algebra A of dimension D s.t. $A/R(A)$ is commutative, and an $n \times n$ matrix M with entries from A, computes the determinant of M in time $N^{O(d)}$, where d is the nilpotency index of $R(A)$ and N is the size of the input.*

In particular, when A is a fixed algebra, $d \leq D = O(1)$, and hence Theorem 5.4 gives us a polynomial time algorithm. This immediately yields the sharp dichotomy theorem in the case of a fixed algebra over finite fields of odd characteristic.

Corollary 5.5. *Let $\mathbb{F}$ be any finite field of odd characteristic and A be any fixed algebra over $\mathbb{F}$. Then, if $A/R(A)$ is noncommutative, computing the determinant over A is* $\mathrm{Mod}_p P$*-hard. If $A/R(A)$ is commutative, then the determinant can be computed in polynomial time.*

Proof of Theorem 5.4. The algorithm first computes the Wedderburn-Malcev decomposition $A = B \oplus' R(A)$ of the algebra A: a result of de Graaf et al. ([9]) shows that such a decomposition may be computed efficiently. By Lemma 5.2, we can compute the nilpotency index d of the algebra in deterministic polynomial time. We assume that $d \leq n$; otherwise, the brute force algorithm for the determinant has running time $N^{O(d)}$.

For any i and j, the (i, j)th entry of the input matrix M can be written uniquely as $m_{i,j} = b_{i,j} + r_{i,j}$ where $b_{i,j} \in B$ and $r_{i,j} \in R(A)$; the elements $b_{i,j}$ and $r_{i,j}$ are also efficiently computable. Now, note that the determinant of the input matrix M can be written as

$$\begin{aligned} \det(M) &= \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} (b_{i,\sigma(i)} + r_{i,\sigma(i)}) \\ &= \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \sum_{S \subseteq [n]} t(\sigma, S) \end{aligned}$$

where $t(\sigma, S)$ is the product, in increasing order of i, of $r_{i,j}$ for $i \in S$ and $b_{i,j}$ for $i \notin S$. Note that $t(\sigma, S) \in R(A)^{|S|}$ (we use here the fact that $R(A)$ is an ideal in A) and hence, $t(\sigma, S) = 0$ if $|S| \geq d$. Thus, we need only consider S of size strictly less than d.

We divide the terms $t(\sigma, S)$ based on the $r_{i,j}$ that actually appear in $t(\sigma, S)$. Specifically, for each 1-1 function $f : S \to [n]$, let $t(\sigma, S, f)$ denote $t(\sigma, S)$ if $\sigma|_S = f$ and 0 otherwise. We can write the determinant $\det(M)$ as

$$\begin{aligned} \det(M) &= \sum_{\substack{S \subseteq [n]: \\ |S| < d}} \sum_{\substack{f : S \to [n]: \\ f\ 1-1}} \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) t(\sigma, S, f) \\ &= \sum_{\substack{S \subseteq [n]: \\ |S| < d}} \sum_{\substack{f : S \to [n]: \\ f\ 1-1}} \det(M(S, f)) \end{aligned}$$

where the entries $m(S, f)_{i,j}$ of $M(S, f)$ are defined as follows: for $i \in S$, $m(S, f)_{i,j} = 0$ if $f(i) \neq j$ and $r_{i,j}$ otherwise; for $i \notin S$, $m(S, f)_{i,j} = b_{i,j}$. We show that for each S and f as above, $\det(M(S, f))$ can be computed in time $N^{O(d)}$, which will prove the theorem since there are only $n^{O(d)}$ of them to compute. For the remainder of the proof, we fix some subset $S \subseteq [n]$ of size $t < d$ and $f : S \to [n]$ that is 1-1.

Note that the matrix $M(S, f)$ is "almost" a matrix over the commutative subalgebra B of A: it contains exactly d entries outside B, one in each row indexed by an element of S. We reduce the computation of $\det(M(S, f))$ to the computation of the determinant of a similar matrix over a commutative algebra closely related to B. Indeed, let $B^{\otimes(t+1)}$ denote $B \otimes B \otimes \cdots \otimes B$ ($t+1$ times). This is a commutative algebra of dimension at most D^d. Furthermore, we see that $1^{\otimes(t+1)}$ is the identity element of this algebra, which we denote by $\mathbb{1}$. For $i \in [t] \cup \{0\}$, we denote by B_i the following subalgebra of $B^{\otimes(t+1)}$: $1^{\otimes i} \otimes B \otimes 1^{\otimes(t-i)}$. It can easily be seen that each B_i is isomorphic to B by the isomorphism $\phi_i : B \to B_i$ where $\phi_i(b) = 1^{\otimes i} \otimes b \otimes 1^{\otimes(t-i)}$.

For $i \in [n]$, we denote by $\mathrm{Pre}(i)$ the set $\{i' \in S \mid i' < i\}$. We now construct a new matrix $M'(S, f)$ with entries from $B^{\otimes(t+1)}$ as follows:

$$m'(S, f)_{i,j} = \begin{cases} 0 & \text{if } i \in S \text{ and } f(i) \neq j, \\ \mathbb{1} & \text{if } i \in S \text{ and } f(i) = j, \\ \phi_\ell(m(S, f)_{i,j}) & \text{if } i \notin S \text{ and } \ell = |\mathrm{Pre}(i)|. \end{cases}$$

In words, to construct $M'(S, f)$ we replace each entry in $M(S, f)$ that is in $R(A)$ by the identity $\mathbb{1}$, and each entry $b_{i,j} \in B$ by the corresponding element in B_ℓ, where $\ell = |\mathrm{Pre}(i)|$.

Since $M'(S, f)$ is a matrix with entries from the commutative algebra $B^{\otimes(t+1)}$, its determinant can be computed in time $N^{O(d)}$. Say $S = \{i_1, \dots, i_t\}$ and $f(i_k) = j_k$ for $k \in [t]$. Let $\{e_1, \dots, e_m\}$ be a basis for B. Then, we have $\det(M(S, f)) = \sum_{\substack{\sigma \in S_n: \\ \sigma|_S = f}} \mathrm{sgn}(\sigma) \left(\prod_{i < i_1} b_{i,\sigma(i)}\right) \cdot r_{i_1,j_1} \cdot \left(\prod_{i_1 < i < i_2} b_{i,\sigma(i)}\right) \cdot r_{i_2,j_2} \cdots r_{i_t,j_t} \cdot \left(\prod_{i > i_t} b_{i,\sigma(i)}\right).$

Each product of the form $\prod_{i \in T} b_{i,\sigma(i)}$ that appears in the summation above is an element of the commutative algebra B and hence $\det(M(S,f))$ can be expanded in the basis of B as follows:

$$\begin{aligned}
&\sum_{\substack{\sigma \in S_n: \\ \sigma|_S = f}} \operatorname{sgn}(\sigma) \left(\sum_{k=1}^{m} \alpha^{(0)}_{k,\sigma} e_k \right) \cdot r_{i_1,j_1} \cdots r_{i_t,j_t} \cdot \left(\sum_{k=1}^{m} \alpha^{(t)}_{k,\sigma} e_k \right) \\
&= \sum_{\overline{k} \in [m]^{t+1}} \left(\sum_{\sigma} \operatorname{sgn}(\sigma) \alpha^{(0)}_{k_0,\sigma} \cdots \alpha^{(t)}_{k_t,\sigma} \right) e_{k_0} r_{i_1,j_1} \cdots r_{i_t,j_t} e_{k_t} \\
&= \sum_{\overline{k} \in [m]^{t+1}} c_{\overline{k}} \cdot e_{k_0} r_{i_1,j_1} e_{k_1} \cdots r_{i_t,j_t} e_{k_t}, \qquad (5.1)
\end{aligned}$$

where $\overline{k}$ denotes the tuple $(k_0, \ldots, k_t)$ and $c_{\overline{k}}$ denotes $\sum_{\sigma:\sigma|_S = f} \alpha^{(0)}_{k_0,\sigma} \cdots \alpha^{(t)}_{k_t,\sigma}$. Let us expand $\det(M'(S,f))$ similarly. We use $e^{(\ell)}_k$ to denote $\phi_\ell(e_k)$. We can write $\det(M'(S,f))$ as

$$\begin{aligned}
&\sum_{\substack{\sigma \in S_n: \\ \sigma|_S = f}} \operatorname{sgn}(\sigma) \prod_{i < i_1} \phi_0(b_{i,\sigma(i)}) \cdot \mathbb{1} \cdots \mathbb{1} \cdot \prod_{i > i_t} \phi_t(b_{i,\sigma(i)}) \\
&= \sum_{\substack{\sigma \in S_n: \\ \sigma|_S = f}} \operatorname{sgn}(\sigma) \left(\sum_{k=1}^{m} \alpha^{(0)}_{k,\sigma} e^{(0)}_k \right) \cdot \mathbb{1} \cdots \mathbb{1} \cdot \left(\sum_{k=1}^{m} \alpha^{(t)}_{k,\sigma} e^{(t)}_k \right) \\
&= \sum_{\overline{k}} c_{\overline{k}} \cdot e^{(0)}_{k_0} e^{(1)}_{k_1} \cdots e^{(t)}_{k_t} = \sum_{\overline{k}} c_{\overline{k}} \cdot e_{k_0} \otimes e_{k_1} \otimes \cdots \otimes e_{k_t}.
\end{aligned}$$

Thus, we can simply read off the coefficients $c_{\overline{k}}$ from $\det(M'(S,f))$ and, using Equation (5.1), we can compute $\det(M(S,f))$. Since $\det(M'(S,f))$ can be computed in time $N^{O(d)}$, we obtain a $N^{O(d)}$-time algorithm to compute $\det(M(S,f))$ and hence $\det(M)$ as well. □

6. DISCUSSION

Our results show that the basic Godsil-Gutman approach to approximating the permanent, as generalized by Chien et al. [5] runs into many obstacles, since the estimators are not efficiently computable. In the case of the quaternions, the result of Chien et al. shows that a suitable modification of the basic estimator still gives a relatively good approximation to the permanent. Is there such a modification for matrix algebras?

Our dichotomy theorem in Section 5 used crucially the fact that we worked over a finite field of odd characteristic. Over the rationals, for example, even the structure of semisimple algebras is fairly complicated, and we do not have an exact characterization of when the determinant over such an algebra is efficiently computable. Extending our dichotomy theorem to these algebras is an interesting open problem.

Theorem 5.4 shows that even when given the algebra A as input, the determinant remains efficiently computable as long as $A/R(A)$ is commutative and A has bounded nilpotency index. How close is this to being a characterization of algebras over which the determinant is polynomial time computable (under reasonable complexity assumptions) when the algebra is part of the input? More generally, can one come up with suitable conditions on the radical $R(A)$ under which computing the determinant over A is hard even when $A/R(A)$ is commutative?

7. REFERENCES

[1] Arora, S., and Barak, B. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[2] Arvind, V., and Srinivasan, S. On the hardness of the noncommutative determinant. In *Proc. 42nd ACM Symp. on Theory of Computing (STOC)* (2010), pp. 677–686.

[3] Barvinok, A. I. Polynomial time algorithms to approximate permanents and mixed discriminants within a simply exponential factor. *Random Struct. Algorithms 14*, 1 (1999), 29–61.

[4] Berkowitz, S. J. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett. 18*, 3 (1984), 147–150.

[5] Chien, S., Rasmussen, L. E., and Sinclair, A. Clifford algebras and approximating the permanent. *J. Computer and System Sciences 67*, 2 (2003), 263–290. (Preliminary version in *34th STOC*, 2002).

[6] Chien, S., and Sinclair, A. Algebras with polynomial identities and computing the determinant. *SIAM J. Computing 37*, 1 (2007), 252–266. (Preliminary version in *45th FOCS*, 2004).

[7] Chistov, A. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Fundamentals of Computation Theory*, L. Budach, Ed., vol. 199 of *LNCS*. Springer, 1985, pp. 63–69.

[8] Curtis, C. W., and Reiner, I. *Representation Theory of Finite Groups and Associative Algebras.* No. XI in Pure and Applied Mathematics. Interscience Publishers, 1962.

[9] de Graaf, W. A., Ivanyos, G., Küronya, A., and Rónyai, L. Computing levi decompositions in lie algebras. *Appl. Algebra Eng. Commun. Comput. 8*, 4 (1997), 291–303.

[10] Godsil, C. D., and Gutman, I. On the matching polynomial of a graph. In *Algebraic Methods in Graph Theory, Vol. I*, L. Lovász and V. T. S'os, Eds. North Holland, Amsterdam–New York, 1981, pp. 241–249.

[11] Hyafil, L. The power of commutativity. In *Proc. 18th IEEE Symp. on Foundations of Comp. Science (FOCS)* (1977), pp. 171–174.

[12] Karmarkar, N., Karp, R. M., Lipton, R. J., Lovász, L., and Luby, M. A monte-carlo algorithm for estimating the permanent. *SIAM J. Computing 22*, 2 (1993), 284–293.

[13] Lam, T.-Y. *A first course in noncommutative rings*, vol. 131 of *Graduate texts in Mathematics.* Springer, 1991.

[14] Mahajan, M., and Vinay, V. Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci. 1997*, 5 (1997). (Preliminary version in *8th SODA*, 1997).

[15] Moore, C., and Russell, A. Approximating the permanent via nonabelian determinants, 2009.

[16] Nisan, N. Lower bounds for non-commutative computation (extended abstract). In *Proc. 23rd ACM Symp. on Theory of Computing (STOC)* (1991), pp. 410–418.

[17] Papadimitriou, C. H. *Computational Complexity.* Addison-Wesley, 1994.

[18] Samuelson, P. A. A method of determining explicitly the coefficients of the characteristic equation. *Annals of Mathematical Statistics 13*, 4 (1942), 424–429.

[19] Valiant, L. G. The complexity of computing the permanent. *Theoretical Comp. Science 8* (1979), 189–201.

[20] Winograd, S. On the number of multiplications necessary to compute certain functions. *Comm. on Pure and Appl. Math. 23*, 2 (1970), 165–179.