Randomized Polynomial-Time Equivalence Between Determinant and Trace-IMM Equivalence Tests

Janaky Murthy
Indian Institute of Science, Bangalore, India
janakymurthy@iisc.ac.in

Vineet Nair [1]
Technion Israel Institute of Technology, Haifa, Israel
vineet@cs.technion.ac.il

Chandan Saha
Indian Institute of Science, Bangalore, India
chandan@iisc.ac.in

Abstract

Equivalence testing for a polynomial family $\{g_m\}_{m\in\mathbb{N}}$ over a field $\mathbb{F}$ is the following problem: Given black-box access to an n-variate polynomial $f(\mathbf{x})$, where n is the number of variables in g_m for some $m \in \mathbb{N}$, check if there exists an $A \in \mathrm{GL}(n,\mathbb{F})$ such that $f(\mathbf{x}) = g_m(A\mathbf{x})$. If yes, then output such an A. The complexity of equivalence testing has been studied for a number of important polynomial families, including the determinant (Det) and the family of iterated matrix multiplication polynomials. Two popular variants of the iterated matrix multiplication polynomial are: $\mathrm{IMM}_{w,d}$ (the $(1,1)$ entry of the product of d many $w \times w$ symbolic matrices) and $\text{Tr-IMM}_{w,d}$ (the trace of the product of d many $w \times w$ symbolic matrices). The families – Det, IMM and Tr-IMM – are VBP-complete under p-projections, and so, in this sense, they have the same complexity. But, do they have the same equivalence testing complexity? We show that the answer is "yes" for Det and Tr-IMM (modulo the use of randomness).

The above result may appear a bit surprising as the complexity of equivalence testing for IMM and that for Det are quite different over $\mathbb{Q}$: a randomized poly-time equivalence testing for IMM over $\mathbb{Q}$ is known [28], whereas [15] showed that equivalence testing for Det over $\mathbb{Q}$ is integer factoring hard (under randomized reductions and assuming GRH). To our knowledge, the complexity of equivalence testing for Tr-IMM was not known before this work. We show that, despite the syntactic similarity between IMM and Tr-IMM, equivalence testing for Tr-IMM and that for Det are randomized poly-time Turing reducible to each other over any field of characteristic zero or sufficiently large. The result is obtained by connecting the two problems via another well-studied problem in computer algebra, namely the *full matrix algebra isomorphism* problem (FMAI). In particular, we prove the following:

1. Testing equivalence of polynomials to $\text{Tr-IMM}_{w,d}$, for $d \geq 3$ and $w \geq 2$, is randomized polynomial-time Turing reducible to testing equivalence of polynomials to Det_w, the determinant of the $w \times w$ matrix of formal variables. (Here, d need not be a constant.)
2. FMAI is randomized polynomial-time Turing reducible to equivalence testing (in fact, to tensor isomorphism testing) for the family of *matrix multiplication tensors* $\{\text{Tr-IMM}_{w,3}\}_{w\in\mathbb{N}}$.

These results, in conjunction with the randomized poly-time reduction (shown in [15]) from determinant equivalence testing to FMAI, imply that the four problems – FMAI, equivalence testing for Tr-IMM and for Det, and the 3-tensor isomorphism problem for the family of matrix multiplication tensors – are randomized poly-time equivalent under Turing reductions.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases equivalence testing, determinant, trace of the matrix product, full-matrix algebra isomorphism

[1] A part of this work was done when the author was still a graduate student at Indian Institute of Science.

45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020).
Editors: Javier Esparza and Daniel Kráľ; Article No. 72; pp. 72:1–72:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.72

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2020/091/`.

Funding *Vineet Nair* : The author is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 682203 -ERC-[Inf-Speed-Tradeoff].

Acknowledgements We are thankful to Avi Wigderson for his suggestion on designing an equivalence testing algorithm for Tr-IMM at the end of VN's presentation at CCC 2017. We would like to thank Christian Ikenmeyer for his question on equivalence testing for Tr-IMM which encouraged us to work on this problem. Thanks also to Neeraj Kayal and Ankit Garg for helpful discussions, and particularly to Neeraj for pointing us to [20]. We thank the anonymous reviewers for their helpful comments.

1 Introduction

The *polynomial equivalence problem* or *equivalence testing* is the following algorithmic task: Given two n-variate polynomials f and g over a field $\mathbb{F}$ as lists of coefficients, determine if there exists an $A \in \mathrm{GL}(n, \mathbb{F})$ such that $f(\mathbf{x}) = g(A\mathbf{x})$. If yes, then f is said to be *equivalent to*[2] g over $\mathbb{F}$. The complexity of equivalence testing depends on the underlying field $\mathbb{F}$. Over finite fields, the problem is in $\mathsf{NP} \cap \mathsf{coAM}$ [45, 48][3], and hence unlikely to be NP-complete. Whereas over $\mathbb{Q}$, it is not even known whether equivalence testing is decidable. The best known complexity of the problem over other fields follows from a naive reduction to solving a system of polynomial equations. However, polynomial solvability could be harder than testing polynomial equivalence.

Connections to other problems. A few works in the literature have related equivalence testing to other fundamental problems. For example, [1] showed that the special instance of cubic form equivalence is at least as hard as (but possibly harder than) graph isomorphism, irrespective of the underlying field. There is a close connection between cubic form equivalence and the algebra isomorphism problem. [2] gave a polynomial-time reduction from commutative algebra isomorphism to cubic form equivalence over any field. In the reverse direction, a polynomial-time reduction is known from cubic form equivalence to commutative algebra isomorphism over *almost* all fields [1, 20]. In fact, the results in [6], [14] and [20] together imply that a host of problems, which includes 3-tensor isomorphism, matrix space isometry, matrix space conjugacy, (commutative or associative) algebra isomorphism and cubic form equivalence, are polynomial-time reducible to each other. There is a cryptographic authentication scheme [40] based on the presumed hardness of cubic form equivalence[4] over finite fields (or rather a generalization of it known as *Isomorphism of Polynomials with one Secret* (IP1S)). It is not known whether cubic form equivalence is even decidable over $\mathbb{Q}$. In contrast, the complexity of quadratic form equivalence testing is completely resolved, primarily due to well-known classification results for quadratic forms (see [3, 46]). The classification yields a polynomial-time quadratic form equivalence testing over finite fields. Over $\mathbb{Q}$ though, quadratic form equivalence can be solved in polynomial time only with oracle access to integer factoring. Moreover, integer factoring reduces in randomized poly-time to the search version of quadratic form equivalence over $\mathbb{Q}$ [50].

[2] Indeed, f and g represent the same function on $\mathbb{F}^n$ upto a change of basis.
[3] This is shown by using the classic set lower bound protocol [18].
[4] more generally, constant-degree form equivalence

Special polynomial families. The work of [24] initiated the study of a natural variant of the polynomial equivalence problem, namely equivalence testing for special families of polynomials. In this setting, we fix some important family of polynomials $\mathcal{G} = \{g_m\}_{m \in N}$ and then aim to design an equivalence testing algorithm for $\mathcal{G}$. Such an algorithm takes input black-box access[5] to a single n-variate polynomial $f(\mathbf{x})$ and determines whether f is equivalent to g_m for some $m \in \mathbb{N}$, and if yes, then it also outputs an $A \in \mathrm{GL}(n, \mathbb{F})$ such that $f(\mathbf{x}) = g_m(A\mathbf{x})$. [24, 25] gave randomized polynomial-time equivalence testing algorithms for a few interesting polynomial families, viz. the determinant, the permanent, the family of elementary symmetric polynomials and the family of power symmetric polynomials. These families are quite popular in algebraic complexity theory, particularly in the context of proving arithmetic circuit lower bounds (see the surveys [8, 44, 47]). Except for the determinant, the algorithms in [24, 25] work over $\mathbb{C}, \mathbb{Q}$, and finite fields[6], and for the determinant it works only over $\mathbb{C}$. Recently, [15] gave a randomized polynomial-time equivalence testing algorithm for the determinant over finite fields[7]. They also showed that determinant equivalence test over $\mathbb{Q}$ is intimately connected to integer factoring: Let $\mathrm{Det}_w(\mathbf{x})$ be the determinant of the $w \times w$ symbolic matrix. Then, deciding if a given polynomial is equivalent to Det_w over $\mathbb{Q}$ can be done in randomized polynomial-time with oracle access to integer factoring, provided w is a constant[8]. Furthermore, assuming GRH, there is a randomized polynomial-time reduction from factoring square-free integers to finding an $A \in \mathrm{GL}(2, \mathbb{Q})$ such that a given quadratic form $f = \mathrm{Det}_2(A\mathbf{x})$, if f is equivalent to Det_2.

Determinant equivalence test is particularly interesting in the context of the permanent versus determinant problem [49]. An approach to solve this long-standing open problem is given by Geometric Complexity Theory (GCT) [35, 36], which proposes the applications of deep tools and techniques from algebraic geometry, group theory and representation theory to achieve this goal. GCT reduces the problem to showing that the (padded) permanent polynomial is not in the *orbit closure*[9] of a polynomial-size determinant polynomial, and suggests (among other things) to develop an algorithmic approach to do the same. Equivalence testing for the determinant is the related problem of checking if a given polynomial is in the orbit of the determinant polynomial.

The determinant Det $:= \{\mathrm{Det}_w\}_{w \in \mathbb{N}}$ is complete (under p-projections) for the class VBP [34] [10]. Likewise, the family of *iterated matrix multiplication* polynomials is also complete for the class VBP, and has been used quite a bit in proving arithmetic circuit lower bounds. In this sense, the two families have the same complexity. But, do they have similar equivalence testing complexity? Our work here, in conjunction with [15] and [28], gives an answer to this question.

---

[5] i.e., query access to evaluations of f at chosen points from $\mathbb{F}^n$.

[6] Over $\mathbb{C}$, the computation model assumes that arithmetic with numbers in $\mathbb{C}$ and root finding of univariate polynomials over $\mathbb{C}$ can be done efficiently. Also, the finite fields are assumed to be of sufficiently large characteristic.

[7] A determinant equivalence test over finite fields was also given in [26], but the algorithm there outputs an invertible transformation over a low extension of the base field.

[8] When w is not a constant, [15] gave a randomized polynomial-time determinant equivalence test over $\mathbb{Q}$, but the algorithm (which works without an integer factoring oracle) outputs a transformation over a low extension of $\mathbb{Q}$.

[9] The orbit of an n-variate degree-d polynomial $g \in \mathbb{C}[\mathbf{x}]$ is the set $\{g(A\mathbf{x}) \mid A \in \mathrm{GL}(n, \mathbb{C})\}$, and the orbit closure of g is the Zariski closure of the orbit when viewed as points in $\mathbb{C}^{\binom{n+d}{d}}$.

[10] Class VBP consists of polynomial families that are computable by polynomial-size algebraic branching programs (ABP). ABP is a powerful model for computing polynomials that subsumes arithmetic formulas.

Iterated matrix multiplication. Two natural versions of the iterated matrix multiplication polynomial are: a) $\mathrm{IMM}_{w,d}$ that is defined as the $(1,1)$ entry of the product of d many $w \times w$ symbolic matrices (i.e., matrices whose entries are distinct variables), and b) $\text{Tr-IMM}_{w,d}$ that is defined as the trace of the product of d many $w \times w$ symbolic matrices. The $\mathrm{IMM} := \{\mathrm{IMM}_{w,d}\}_{w,d \in \mathbb{N}}$ family has been studied more from the lower bound perspective [9, 12, 27, 29, 30, 32, 39] because it naturally captures the algebraic branching program model. On the other hand, $\text{Tr-IMM} := \{\text{Tr-IMM}_{w,d}\}_{w,d \in \mathbb{N}}$ has been studied in [16, 17, 19, 33][11] owing to its nice structural properties (pertaining to its group of symmetries and the associated Lie algebra) that may be quite useful for studying GCT methods when applied to the "Permanent versus Tr-IMM" problem. IMM and Tr-IMM are also complete for the class VBP. Interestingly, the three polynomials – Det_w, $\mathrm{IMM}_{w,d}$ and $\text{Tr-IMM}_{w,d}$ – are characterized by their respective groups of symmetries [13, 16, 28].

Equivalence testing for iterated matrix multiplication. How does equivalence testing for IMM and Tr-IMM relate to that of Det? In [28], a randomized polynomial-time equivalence testing algorithm was given for IMM over $\mathbb{C}, \mathbb{Q}$ and finite fields. Comparing this with the above-mentioned results on determinant equivalence test [15, 25], we see that the complexity of equivalence tests for Det and IMM are quite different over $\mathbb{Q}$ (unless integer factoring is easy). Is this also the case between Det and Tr-IMM? One may be tempted to say "yes" owing to the closeness of the definitions of IMM and Tr-IMM. However, contrary to this first impression, we show that equivalence testing for Det and that for Tr-IMM are randomized polynomial-time Turing reducible to each other over $\mathbb{C}$, $\mathbb{Q}$ and finite fields[12] (see Corollary 3). Thus, viewed in this way, Det and Tr-IMM are closer to each other than to IMM.[13] For brevity, we would henceforth denote the equivalence testing problems for Det and Tr-IMM by DET and TRACE respectively.

Connections to algebra isomorphism and 3-tensor isomorphism. As mentioned before, cubic form equivalence, algebra isomorphism and 3-tensor isomorphism are polynomial-time equivalent. Moreover, degree-d form equivalence reduces to cubic form equivalence [1, 2] and d-tensor isomorphism reduces to 3-tensor isomorphism [20] in polynomial-time, if d is bounded. Det and Tr-IMM being two important polynomial families, we wonder if DET and TRACE can be linked with any natural case of algebra isomorphism. Further, do DET and TRACE reduce to any special case of cubic form equivalence or 3-tensor isomorphism? We show that the answers to these are "yes". The relevant problems are the *full-matrix algebra isomorphism* (FMAI) problem and the 3-tensor isomorphism problem for the family of *matrix multiplication tensors* (MMTI).

FMAI is a well-studied problem in computer algebra which is defined as follows: Given a basis of a matrix algebra $\mathcal{A} \subseteq \mathcal{M}_m(\mathbb{F})$, check if $\mathcal{A}$ is isomorphic[14] to $\mathcal{M}_w(\mathbb{F})$, where $\mathcal{M}_m(\mathbb{F})$ is the algebra of $m \times m$ matrices over $\mathbb{F}$ and $\dim_{\mathbb{F}}(\mathcal{A}) = w^2$, and if yes then output an isomorphism from $\mathcal{A}$ to $\mathcal{M}_w(\mathbb{F})$. A randomized polynomial-time algorithm to solve FMAI

[11] Actually, [17] studied a related polynomial $\text{Tr-Pow}_{w,d}$, which is the trace of the d-th power of a $w \times w$ symbolic matrix. They showed that a particular line of attack prescribed by GCT, namely *orbit occurrence obstructions*, cannot prove super-linear lower bound on the "Tr-Pow complexity" of the permanent. We are not aware of a similar result (or, more generally, a result that rules out the *occurrence obstructions* approach as in [7, 21]) with Tr-Pow (or Det) replaced by Tr-IMM.

[12] The reduction works over any adequately large field $\mathbb{F}$ of characteristic zero or sufficiently large. We also require that univariate polynomial factoring over $\mathbb{F}$ can be done efficiently.

[13] Talking of the difference between the "trace model" and the "(1,1) model", a recent work [5] showed that in the non-commutative setting, the border width complexity and the width complexity of a polynomial are *not* always equal for the trace-ABP model, unlike the case for the classical $(1,1)$-ABP model [38].

[14] i.e., isomorphic as algebras over $\mathbb{F}$.

over finite fields was given in [41, 42], whereas over $\mathbb{Q}$ a randomized Turing reduction from FMAI to integer factoring was shown in [10, 22]. The reduction is polynomial-time if $\dim_{\mathbb{Q}}(\mathcal{A})$ is bounded. Also, [4, 11] gave a randomized polynomial-time algorithm that outputs an isomorphism from $\mathcal{A} \otimes_{\mathbb{Q}} \mathbb{L}$ to $\mathcal{M}_w(\mathbb{L})$, where $\mathbb{L}$ is a degree w extension field of $\mathbb{Q}$, if $\mathcal{A}$ is isomorphic to $\mathcal{M}_w(\mathbb{Q})$. The decision version of FMAI over $\mathbb{Q}$ is in NP $\cap$ coNP [43]. The results for DET in [15] were obtained by giving a randomized poly-time Turing reduction from DET to FMAI. In this work, we give a randomized polynomial-time Turing reduction from TRACE to DET (Theorem 1).

A d-tensor is a degree-d form (i.e., a degree-d homogeneous polynomial) $f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$ whose every monomial has exactly one variable from each of the sets $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d$. The d-tensor isomorphism problem is the following: Given two d-tensors $f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$ and $g(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$ decide if there exist $A_1 \in \mathrm{GL}(|\mathbf{x}_1|, \mathbb{F}), \ldots, A_d \in \mathrm{GL}(|\mathbf{x}_d|, \mathbb{F})$ such that $f = g(A_1\mathbf{x}_1, A_2\mathbf{x}_2, \ldots, A_d\mathbf{x}_d)$. The d-tensor isomorphism problem for a family of d-tensors is defined accordingly, just like equivalence testing for a family of polynomials. MMTI is the 3-tensor isomorphism problem for the family of matrix multiplication tensors $\{\text{Tr-IMM}_{w,3}\}_{w \in \mathbb{N}}$. The matrix multiplication tensor $\text{Tr-IMM}_{w,3}$ is a crucial object in the study of asymptotically fast algorithms for multiplying two $w \times w$ matrices. In this paper, we give a randomized polynomial-time Turing reduction from FMAI to MMTI (Theorem 2). Further, it follows easily from the symmetries of $\text{Tr-IMM}_{w,d}$ ([16], see Lemma 14) that MMTI reduces in polynomial-time to TRACE.

Thus, the above results together with the reduction in [15] show that the four problems – TRACE, DET, FMAI and MMTI – are randomized polynomial-time Turing reducible to each other. Although, the equivalence between MMTI and FMAI has the same essence as the equivalence between 3-tensor isomorphism (or cubic form equivalence) and algebra isomorphism, our proofs are quite different from the proofs in [1, 2, 14, 20][15]. In particular, we do not see any easy adaptation of the arguments in [1, 2, 14, 20] leading to the results mentioned above. Our proofs link MMTI with FMAI, via TRACE and DET, by exploiting the structure of the Lie algebra of $\text{Tr-IMM}_{w,d}$ (which is in the same spirit as the reduction from DET to FMAI in [15] using the Lie algebra of Det_w). Also, the reduction from d-tensor isomorphism (similarly, degree-d form equivalence) to 3-tensor isomorphism (respectively, cubic form equivalence) in [1, 2, 20] is efficient only if d is a constant. Whereas, our reduction from testing equivalence to $\text{Tr-IMM}_{w,d}$ to MMTI runs in time $\mathrm{poly}(w, d)$.

1.1 The results (stated formally)

The polynomial $\text{Tr-IMM}_{w,d} := \mathrm{tr}(Q_0 \cdot Q_1 \ldots Q_{d-1})$, where Q_k is a $w \times w$ symbolic matrix in $\mathbf{x}_k$ variables. Throughout, we will assume that $w \geq 2$, $d \geq 3$ and $\mathrm{char}(\mathbb{F}) = 0$ or $> (w^2 d)^5$, and univariate polynomial factoring over $\mathbb{F}$ can be done in probabilistic polynomial time. The restriction on the characteristic of $\mathbb{F}$ has not been optimized in this paper. The missing proofs can be found in the extended version of this paper (see [37]).

▶ **Theorem 1** (TRACE to DET). *There is a randomized algorithm that takes as input black-box access to an n-variate degree-d polynomial f and oracle access to* DET *over $\mathbb{F}$, and does the following with high probability: If there is a $w \in \mathbb{N}$ such that f is equivalent to Tr-IMM$_{w,d}$, then it outputs an $A \in GL(n, \mathbb{F})$ such that $f =$ Tr-IMM$_{w,d}(A\mathbf{x})$, otherwise it outputs "No such w exists". The algorithm runs in* $\mathrm{poly}(n, \beta)$ *time, where β is the bit length of the coefficients of f.*

[15] The reductions in these prior works are deterministic and hold for the decision versions of the problems, whereas the reductions here are randomized and for the search versions of the problems.

The reduction is given in Section 4. Theorem 1 implies a randomized poly-time algorithm for TRACE over $\mathbb{C}$ and finite fields, and also over $\mathbb{Q}$ (provided the algorithm has access to integer factoring oracle and w is bounded) via known results on DET [15, 25]. Two other remarks:

1. *No knowledge of w*: The algorithm requires no knowledge of w, if the input polynomial f is equivalent to $\text{Tr-IMM}_{w,d}$ for some $w \in \mathbb{N}$ then the algorithm finds such a w.
2. *Reduction to* TRACE-TI: The *tensor isomorphism* problem for Tr-IMM (denoted TRACE-TI) is as follows: Given blackbox access to a d-tensor $g(\mathbf{x}_0, \ldots, \mathbf{x}_{d-1})$, check if there are $B_0, \ldots, B_{d-1} \in \text{GL}(w^2, \mathbb{F})$ such that $g = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$, and if yes then output such $B_0, \ldots, B_{d-1}$. The algorithm in Theorem 1 first reduces TRACE to TRACE-TI (finding w in this step), and then solves TRACE-TI using DET oracle over $\mathbb{F}$. The reduction from TRACE to TRACE-TI (which resembles a similar reduction used in the equivalence test for IMM [28]) does not require oracle access to DET. A randomized polynomial-time algorithm for TRACE-TI *over* $\mathbb{C}$ was given in [19], but the algorithm does not reduce TRACE-TI to DET.

▶ **Theorem 2** (FMAI to MMTI)**.** *There is randomized algorithm that takes as input a basis of an algebra $\mathcal{A} \subseteq \mathcal{M}_m(\mathbb{F})$, and oracle access to* MMTI*, and does the following with high probability: If $\mathcal{A} \cong \mathcal{M}_w(\mathbb{F})$, where $w^2 = \dim_{\mathbb{F}}(\mathcal{A})$ then it outputs 'Yes' and otherwise it outputs 'No such $w \in \mathbb{N}$ exists'. If the algorithm outputs 'Yes' then it also outputs an algebra isomorphism from $\mathcal{A}$ to $\mathcal{M}_w(\mathbb{F})$. The algorithm runs in* $\text{poly}(m, \beta)$ *time, where β is the bit length of the entries of the input basis matrices.*

The algorithm is given in Section 5.2. It uses a characterization of $\text{Tr-IMM}_{w,d}$ by the Lie algebra $\mathfrak{g}_{\text{Tr-IMM}}$ of its group of symmetries (Lemma 17) along with a nice choice of basis of $\mathfrak{g}_{\text{Tr-IMM}}$ (Section 3) to reduce FMAI to degree four TRACE-TI in *deterministic* polynomial time, which in turn reduces to MMTI in randomized polynomial time (Theorem 4). Two more remarks on Theorem 2:

1. MMTI *to* TRACE: Using oracle access to TRACE, it is easy to solve MMTI (in fact TRACE-TI) in polynomial time: Since a polynomial identity test at the end of a TRACE-TI algorithm ensures that the output of the algorithm is correct, it suffices to prove that if the input to a TRACE algorithm is a d-tensor f that is isomorphic to $\text{Tr-IMM}_{w,d}$, then the algorithm outputs d matrices $B_0, \ldots, B_{d-1}$ such that $f(\mathbf{x}) = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$. This is true as any algorithm for TRACE outputs a block-diagonal matrix B such that $f(\mathbf{x}) = \text{Tr-IMM}_{w,d}(B\mathbf{x})$ (from Lemma 14). Matrices $B_0, \ldots, B_{d-1}$ can be easily derived from B.
2. *A reduction from* FMAI *to* DET: A Turing reduction from FMAI to DET over $\mathbb{F}$ was given in [15] that runs in exponential time. We improve this run-time significantly: Theorems 1 and 2 imply that FMAI is in fact randomized polynomial-time Turing reducible to DET.

▶ **Corollary 3.** *It follows from Theorems 1 and 2, and the randomized polynomial-time Turing reduction from* DET *to* FMAI *in [15], that the four problems –* TRACE*,* DET*,* FMAI *and* MMTI *– are randomized polynomial-time equivalent under Turing reductions.*

As mentioned before, the next theorem (proof is omitted) is used in the proof of Theorem 2.

▶ **Theorem 4** (TRACE-TI to MMTI)**.** *There is a randomized algorithm that takes as input black-box access to an n-variate d-tensor $f(\mathbf{x}_0, \ldots, \mathbf{x}_{d-1})$, and oracle access to* MMTI*, and does the following with high probability: If f is isomorphic to $\text{Tr-IMM}_{w,d}$ then it outputs $B_0, B_1, \ldots, B_{d-1} \in GL(w^2, \mathbb{F})$ such that $f = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$, otherwise it outputs "No". The algorithm runs in* $\text{poly}(n, \beta)$ *time, where β is the bit length of the coefficients of f.*

2 Notations and definitions

Recall that $\text{Tr-IMM}_{w,d} := \text{tr}(Q_0 \cdot Q_1 \ldots Q_{d-1})$, where $Q_k = (x_{ij}^{(k)})_{i,j\in[w]}$. Let $\mathbf{x}_k = \{x_{ij}^{(k)}\}_{i,j\in[w]}$, $\mathbf{x} = \uplus_{k\in[0,d-1]}\mathbf{x}_k$, and $n = w^2d$. At times, we will refer to the $\mathbf{x}$ variables as $x_1, \ldots, x_n$. The $\mathbf{x}$ variables are ordered as $\mathbf{x}_0 > \mathbf{x}_1 > \ldots > \mathbf{x}_{d-1}$, and within a variable set $\mathbf{x}_k$, if k is even (similarly, odd) then the variables are ordered in row-major (respectively, column-major) fashion. The rows and columns of a matrix in $\mathcal{M}_n = \mathcal{M}_n(\mathbb{F})$, and the entries of a column vector in $\mathbb{F}^n$ are indexed by $\mathbf{x}$ variables ordered as above. A matrix in $\mathcal{M}_n$ is called *block-diagonal* if the row and column of every non-zero entry of the matrix is indexed by variables from the same variable set. A few more basic definitions and terminologies about matrices, matrix products and ABP (like full-rank linear matrices and matrix products) can be found in [37]. The indices $k, \ell \in [0, d-1]$ will be treated as elements in $\mathbb{Z}/d\mathbb{Z}$, i.e., $k + 1 = 0$ if $k = d - 1$. Let $\mathcal{L} \subseteq \mathcal{M}_n$. A subspace $\mathcal{U} \subseteq \mathbb{F}^n$ is *$\mathcal{L}$-invariant* if for all $M \in \mathcal{L}$, $M \cdot \mathcal{U} \subseteq \mathcal{U}$.

▶ **Definition 5** (Irreducible invariant subspace). *An $\mathcal{L}$-invariant subspace $\mathcal{U} \subseteq \mathbb{F}^n$ is irreducible if there are no proper $\mathcal{L}$-invariant subspaces $\mathcal{U}_1$ and $\mathcal{U}_2$ of $\mathcal{U}$ such that $\mathcal{U} = \mathcal{U}_1 \oplus \mathcal{U}_2$.*

▶ **Definition 6** (Closure of a vector). *The closure of a vector $\mathbf{v} \in \mathbb{F}^n$ under the action of $\mathcal{L} \subseteq \mathcal{M}_n$ is the smallest $\mathcal{L}$-invariant subspace of $\mathbb{F}^n$ containing $\mathbf{v}$.*

An algorithm to compute the closure of a vector in polynomial-time is given in [28]. An easy-to-work-with definition of the Lie algebra of the group of symmetries of a polynomial was given in [25]. For brevity, we will call it the Lie algebra of a polynomial.[16]

▶ **Definition 7** (Lie algebra $\mathfrak{g}_f$ of a polynomial f). *The Lie algebra of an n-variate polynomial $f(\mathbf{x})$ is denoted as $\mathfrak{g}_f$ and it consists of matrices $E = (e_{ij})_{i,j\in[n]} \in \mathcal{M}_n$ that satisfy $\sum_{i,j\in[n]} e_{ij}x_j \cdot \frac{\partial f}{\partial x_i} = 0$.*

Note that $\mathfrak{g}_f$ is a vector space, and a basis of $\mathfrak{g}_f$ can be computed in randomized polynomial-time from blackbox access to f by solving a linear system (see [25]).

▶ **Fact 1.** *If $f(\mathbf{x}) = g(A\mathbf{x})$ for an $A \in GL(n, \mathbb{F})$, then $\mathfrak{g}_f = A^{-1}\mathfrak{g}_g A$.*

3 Symmetries and Lie algebra of Tr-IMM

The symmetries and the Lie algebra $\mathfrak{g}_{\text{Tr-IMM}}$ of $\text{Tr-IMM}_{w,d}$ have been studied in [16] over $\mathbb{C}$. Here, we work out the exact structure of the matrices in $\mathfrak{g}_{\text{Tr-IMM}}$ with respect to the variable ordering mentioned above, and use it to identify the $\mathfrak{g}_{\text{Tr-IMM}}$-invariant subspaces of $\mathbb{F}^n$ and the symmetries of $\text{Tr-IMM}_{w,d}$ over $\mathbb{F}$. These facts about the Lie algebra and the symmetries will be used in the proofs of Theorems 1, 2 and 4.

▷ Claim 8. If $E \in \mathfrak{g}_{\text{Tr-IMM}}$ then E is block-diagonal.

Define the spaces $\mathcal{B}_0, \ldots, \mathcal{B}_{d-1}$ of block-diagonal matrices as follows: Every matrix in $\mathcal{B}_k$ is a block-diagonal matrix whose non-zero entries are confined to the rows and columns indexed by $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ variables. For $k \in [0, d-1]$ and $B \in \mathcal{B}_k$, let $[B]_k$ be the $2w^2 \times 2w^2$

[16] Geometrically speaking, the Lie algebra of an n-variate polynomial $f(\mathbf{x})$ is the subspace of $\mathcal{M}_n(\mathbb{F})$ obtained by translating the tangent of the algebraic set $\{A \in \mathcal{M}_n : f(A\mathbf{x}) = f(\mathbf{x})\}$ at $A = I_n$ and making it pass through origin.

sub-matrix of B whose rows and columns are indexed by $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ variables. If d is even then

$$\mathcal{B}_k := \left\{ B \in \mathcal{M}_n \;:\; [B]_k = \begin{bmatrix} I_w \otimes M^T & \mathbf{0} \\ \mathbf{0} & -I_w \otimes M \end{bmatrix} \text{ for } M \in \mathcal{M}_w \right\} \text{ if } k \text{ is even,}$$
$$\mathcal{B}_k := \left\{ B \in \mathcal{M}_n \;:\; [B]_k = \begin{bmatrix} M^T \otimes I_w & \mathbf{0} \\ \mathbf{0} & -M \otimes I_w \end{bmatrix} \text{ for } M \in \mathcal{M}_w \right\} \text{ if } k \text{ is odd.} \quad (1)$$

If d is odd, then the definition of $\mathcal{B}_k$ remains the same except for $\mathcal{B}_{d-1}$ which is defined as

$$\mathcal{B}_{d-1} := \left\{ B \in \mathcal{M}_n \;:\; [B]_{d-1} = \begin{bmatrix} I_w \otimes M^T & \mathbf{0} \\ \mathbf{0} & -M \otimes I_w \end{bmatrix} \text{ for } M \in \mathcal{M}_w \right\}.$$

▶ **Lemma 9.** *The space $\mathcal{B}_0 + \ldots + \mathcal{B}_{d-1}$ is contained in $\mathfrak{g}_{\textit{Tr-IMM}}$.*

▶ **Lemma 10.** *Suppose $B \in \mathfrak{g}_{\textit{Tr-IMM}}$ and there is a $k \in [0, d-1]$ such that the non-zero entries of B are confined to the rows and columns that are indexed by $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ variables. Then $B \in \mathcal{B}_k$.*

In fact $\mathfrak{g}_{\text{Tr-IMM}} = \mathcal{B}_0 + \ldots + \mathcal{B}_{d-1}$, but we do not prove this stronger statement here. Let $e_i \in \mathbb{F}^n$ be the vector with 1 in the entry indexed by $x_i \in \mathbf{x}$ and zero elsewhere. A subspace of $\mathbb{F}^n$ is a coordinate subspace if it is spanned by a set of e_i's. Let $\mathcal{U}_k = \text{span}_{\mathbb{F}}\{e_i \mid x_i \in \mathbf{x}_k\}$.

▷ Claim 11. Any non-zero $\mathfrak{g}_{\text{Tr-IMM}}$-invariant subspace is a coordinate subspace of $\mathbb{F}^n$.

▶ **Lemma 12.** *The only irreducible $\mathfrak{g}_{\textit{Tr-IMM}}$-invariant subspaces of $\mathbb{F}^n$ are $\mathcal{U}_0, \ldots, \mathcal{U}_{d-1}$.*

▶ **Corollary 13.** *If $f = \textit{Tr-IMM}_{w,d}(A\mathbf{x})$, where $A \in GL(n, \mathbb{F})$, then the only irreducible $\mathfrak{g}_f$-invariant subspaces of $\mathbb{F}^n$ are $A^{-1}\mathcal{U}_0, \ldots, A^{-1}\mathcal{U}_{d-1}$.*

The above lemmas help us derive the group of symmetries of Tr-IMM$_{w,d}$ over $\mathbb{F}$ (proof omitted).

▶ **Lemma 14.** *Let $\textit{Tr-IMM}_{w,d} = \text{tr}(Q'_0 \cdots Q'_{d-1})$, where $Q'_0 \cdots Q'_{d-1}$ is a full-rank (w, d, n)-matrix product over $\mathbb{F}$. Then there are $C_k \in GL(w, \mathbb{F})$ for $k \in [0, d-1]$, and $\ell \in [0, d-1]$ such that either $Q'_k = C_k \cdot Q_{\ell+k} \cdot C_{k+1}^{-1}$ for $k \in [0, d-1]$ or $Q'_k = C_k \cdot Q_{\ell-k}^T \cdot C_{k+1}^{-1}$ for $k \in [0, d-1]$.*

4 Reduction from TRACE to DET: Proof of Theorem 1

The reduction is given in Algorithm 1. The algorithm proceeds by assuming that the input polynomial f is equivalent to Tr-IMM$_{w,d}$ for some $w \geq 2$. A final polynomial identity test (PIT) takes care of the case when it is not. Algorithm 1 has two main steps – reduction from TRACE to TRACE-TI (Algorithm 4 in [37]), and reduction from TRACE-TI to DET (Algorithm 2). The reduction from TRACE to TRACE-TI is inspired by a similar reduction in [28] for the IMM polynomial. Below we discuss the proof strategy of the reduction from TRACE to TRACE-TI, and give the details in the extended version. Algorithm 2 is given in Section 4.1.

Reduction from TRACE to TRACE-TI. First, we compute bases of the irreducible $\mathfrak{g}_f$-invariant subspaces of $\mathbb{F}^n$. By Corollary 13, these are bases of the spaces $A^{-1}\mathcal{U}_{\sigma(0)}, \ldots, A^{-1}\mathcal{U}_{\sigma(d-1)}$, where σ is an unknown permutation on $\{0, \ldots, d-1\}$. As $\dim_{\mathbb{F}}(\mathcal{U}_k) = w^2$, we get w. Now, let V_k be the $n \times w^2$ matrix consisting of the basis vectors of $A^{-1}\mathcal{U}_{\sigma(k)}$. Form the $n \times n$ matrix $V = [V_0 \mid V_1 \mid \ldots \mid V_{d-1}]$. Observe that $V = A^{-1} \cdot E$, where E is a "block-permuted" invertible matrix (by the definition of $\mathcal{U}_k$). Thus, $h(\mathbf{x}) := f(V\mathbf{x}) = \text{Tr-IMM}_{w,d}(E\mathbf{x})$. We now make use of the evaluation dimension measure (Definition C.1 in [37]) on h to essentially ensure that E is a block-diagonal matrix.

Algorithm 1 Reduction from TRACE to DET.

INPUT: Blackbox access to an n-variate, degree d polynomial f and oracle access to DET.
OUTPUT: If there is an $w \in \mathbb{N}$ such that f is equivalent to $\text{Tr-IMM}_{w,d}$ then output an $A \in \text{GL}(n, \mathbb{F})$ such that $f(\mathbf{x}) = \text{Tr-IMM}_{w,d}(A\mathbf{x})$. Otherwise output "No such w exists".

Reduction to TRACE-TI

1: Use Algorithm 4 in [37] with input f to compute $A' \in \text{GL}(n, \mathbb{F})$ and a $w \in \mathbb{N}$ such that $h(\mathbf{x}) = f(A'\mathbf{x})$ is a d-tensor in the variable sets $\mathbf{x}_0, \ldots, \mathbf{x}_{d-1}$ which is isomorphic to $\text{Tr-IMM}_{w,d}$. If the algorithm outputs 'No', output "No such w exists".

Reduction from TRACE-TI to DET

2: Use Algorithm 2 with input h, w and oracle access to DET to compute matrices $B_0, \ldots, B_{d-1} \in \text{GL}(w^2, \mathbb{F})$ such that $h(\mathbf{x}) = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$. If Algorithm 2 outputs 'No' then output "No such w exists".
3: Let $B \in \text{GL}(n, \mathbb{F})$ be the block-diagonal matrix whose k-th block is B_k, and let $A = B(A')^{-1}$.

Final PIT

4: Pick a random point $\mathbf{a} \in S^n$ where $S \subseteq \mathbb{F}$ is of size n^5. If $f(\mathbf{a}) = \text{Tr-IMM}_{w,d}(A\mathbf{a})$ then output w and A, else output "No such w exists".

4.1 Reduction from TRACE-TI to DET

The following two claims (proofs are omitted) help in the argument.

▷ Claim 15. Let X be a $w \times w$ full-rank linear matrix[17] and $Y = I_w \otimes X$. Then there do not exist non-zero matrices $T, S \in \mathcal{M}_{w^2}(\mathbb{F})$ such that $T \cdot Y = Y^T \cdot S$.

▷ Claim 16. Let X be a $w \times w$ full-rank linear matrix and $Y = I_w \otimes X$, and suppose $T, S \in \mathcal{M}_{w^2}(\mathbb{F})$ such that $T \cdot Y = Y \cdot S$. Then $T = S = M \otimes I_w$ for some $M \in \mathcal{M}_w(\mathbb{F})$.

The correctness of Algorithm 2 is argued below by tracing its steps.

Steps 1–3: Assume that h is isomorphic to $\text{Tr-IMM}_{w,d}$. Hence, there is a full-rank (w, d, n) set-multilinear matrix product $X_0 \ldots X_{d-1}$ in $\mathbf{x}_0, \ldots, \mathbf{x}_{d-1}$ variables such that

[17] The entries in a full-rank linear matrix are linearly independent linear forms, and it is different from a matrix whose entries are linear forms and has full-rank over the corresponding field. Note that a full-rank linear matrix has full-rank over the corresponding field but the vice-versa is not always true.

Algorithm 2 Reduction from TRACE-TI to DET.

INPUT: A $w \in \mathbb{N}$, blackbox access to d-tensor $h(\mathbf{x}_0, \dots, \mathbf{x}_{d-1})$ that is isomorphic to Tr-IMM$_{w,d}$, and oracle access to DET.

OUTPUT: Matrices $B_0, \dots, B_{d-1} \in \mathrm{GL}(w^2, \mathbb{F})$ such that $h(\mathbf{x}) = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \dots, B_{d-1}\mathbf{x}_{d-1})$.

1: Use the set-multilinear ABP reconstruction algorithm (which follows from [31]) to construct a (w^2, d, n) set-multilinear ABP $Y_0' \dots Y_{d-1}'$ in $\mathbf{x}_0, \dots, \mathbf{x}_{d-1}$ variables that computes h.

2: For $k \in [1, d-2]$, use the factorization algorithm in [23] to compute blackbox access to a degree-w polynomial g_k such that $\det(Y_k') = \alpha_k g_k(\mathbf{x}_k)^w$, where $\alpha_k \in \mathbb{F}^\times$.

3: For $k \in [1, d-2]$, use the DET oracle on input g_k to compute X_k' such that $\det(X_k') = g_k$. If DET returns g_k is not equivalent to Det_w, then output "No".

4: For $k \in [1, d-2]$, let $Z_k = I_w \otimes X_k'$.

5: For $k \in [1, d-2]$, compute $T_{k-1}', \; S_k' \in \mathrm{GL}(w^2, \mathbb{F})$ such that either $T_{k-1}' \cdot Y_k' = Z_k \cdot S_k'$ or $T_{k-1}' \cdot Y_k' = Z_k^T \cdot S_k'$. If both equalities are satisfied, output "No" (see Observation 4.1).

6: Let $\widehat{Y}_0 = Y_0' \cdot (T_0')^{-1}$, $\widehat{Y}_k = (T_{k-1}') \cdot Y_k' \cdot (T_k')^{-1}$ for $k \in [1, d-3]$, $\widehat{Y}_{d-2} = (T_{d-3}') \cdot Y_{d-2}' \cdot (S_{d-2}')^{-1}$, and $\widehat{Y}_{d-1} = S_{d-2}' \cdot Y_{d-1}'$.

7: Let $\widehat{X}_{d-2}$ be such that $\widehat{Y}_{d-2} = I_w \otimes \widehat{X}_{d-2}$, and for $k \in [1, d-3]$ construct $\widehat{M}_k \in \mathrm{GL}(w, \mathbb{F})$ and $\widehat{X}_k$ such that $\widehat{Y}_k = (\widehat{M}_k \otimes I_w) \cdot (I_w \otimes \widehat{X}_k)$. (See Observation 4.3.)

8: Let $\overline{Y}_{d-1} = (\prod_{k=1}^{d-3} (\widehat{M}_k \otimes I_w)) \cdot \widehat{Y}_{d-1}$. Construct $\widehat{X}_{d-1}$ such that its (i,j)-th entry is the $((j-1)w + i)$-th entry of $\overline{Y}_{d-1}$, and $\widehat{X}_0$ such that its (i,j)-th entry is the $((i-1)w + j)$-th entry of $\widehat{Y}_0$.

9: Obtain the transformations $B_0, \dots, B_{d-1} \in \mathrm{GL}(w^2, \mathbb{F})$ from (the entries of) $\widehat{X}_0, \dots, \widehat{X}_{d-1}$ respectively. Return $B_0, \dots, B_{d-1}$.

$h = \mathrm{tr}(X_0 \dots X_{d-1})$. Observe that, h is computed by the (w^2, d, n)-set-multilinear ABP $Y_0 \dots Y_{d-1}$, where

$$
\begin{aligned}
Y_0 &= (X_0(1,1), \dots, X_0(1,w), X_0(2,1), \dots, X_0(2,w), \dots, X_0(w,1), \dots, X_0(w,w)) \\
Y_k &= I_w \otimes X_k \qquad \text{for } k \in [1, d-2] \\
Y_{d-1} &= (X_{d-1}(1,1), \dots, X_{d-1}(w,1), X_{d-1}(1,2), \dots, X_{d-1}(w,2), \dots, X_{d-1}(1,w), \dots, X_{d-1}(w,w))^T \; .
\end{aligned}
$$

Using the randomized polynomial-time set-multilinear ABP reconstruction algorithm in [31], a (w^2, d, n) set-multilinear ABP $Y_0' \dots Y_{d-1}'$ computing h is constructed in Step 1. It follows from the properties of this algorithm and the ABP $Y_0 \dots Y_{d-1}$ that there are $T_0, \dots, T_{d-2} \in \mathrm{GL}(w^2, \mathbb{F})$ so that

$$
Y_0' = Y_0 \cdot T_0 \; , \qquad Y_k' = T_{k-1}^{-1} \cdot Y_k \cdot T_k \quad \text{for } k \in [1, d-2], \text{ and} \qquad Y_{d-1}' = T_{d-2}^{-1} \cdot Y_{d-1} \; .
$$

Hence, for all $k \in [1, d-2]$, $\det(Y_k') = c_k (\det(X_k))^w$, where $c_k \in \mathbb{F}^\times$. As the determinant polynomial is irreducible, at Step 2, we have $g_k = \beta_k \det(X_k) = \det(\mathrm{diag}(\beta_k, 1, \dots, 1) \cdot X_k)$ for some $\beta_k \in \mathbb{F}^\times$ which implies g_k is equivalent to Det_w. At step 3, DET on input g_k returns X_k' such that

$$
X_k = C_k \cdot X_k' \cdot D_k \quad \text{or} \quad X_k = C_k \cdot (X_k')^T \cdot D_k \qquad \text{where } C_k, D_k \in \mathrm{GL}(w, \mathbb{F}).
$$

The above follows from the group of symmetries of Det_w (see Fact 1 in [26]).

Steps 4–5: At Step 4, for $k \in [1, d-2]$, the matrix $Z_k = I_w \otimes X'_k$ satisfies

$$Y_k = (I_w \otimes C_k) \cdot Z_k \cdot (I_w \otimes D_k) \quad \text{or} \quad Y_k = (I_w \otimes C_k) \cdot Z_k^T \cdot (I_w \otimes D_k).$$

Hence, at Step 5 there are $T'_{k-1} := (I_w \otimes C_k^{-1}) \cdot T_{k-1}$ and $S'_k := (I_w \otimes D_k) \cdot T_k$ in $\mathrm{GL}(w^2, \mathbb{F})$ such that

$$T'_{k-1} \cdot Y'_k = Z_k \cdot S'_k \quad \text{or} \quad T'_{k-1} \cdot Y'_k = Z_k^T \cdot S'_k.$$

Observation 4.1 uses Claim 15 to show that at Step 5 we can identify between the above two cases, as only one of them is true.

▶ **Observation 4.1.** *If $h(\mathbf{x}_0, \ldots, \mathbf{x}_{d-1})$ is isomorphic to Tr-IMM$_{w,d}$ then for matrices Y'_k and Z_k as computed in Algorithm 2, where $k \in [1, d-2]$, there are no matrices $T'_{k-1}, S'_k \in GL(w^2, \mathbb{F})$ such that both $T'_{k-1} \cdot Y'_k = Z_k \cdot S'_k$ and $T'_{k-1} \cdot Y'_k = Z_k^T \cdot S'_k$ are simultaneously true.*

At step 5 the matrices T'_{k-1} and S'_k are computed by solving linear equations. Choosing a solution at random from the solution space ensures that the computed matrices T'_{k-1} and S'_k are invertible with high probability. Henceforth, we assume that $T'_{k-1} \cdot Y'_k = Z_k \cdot S'_k$. The proof for $T'_{k-1} \cdot Y'_k = Z_k^T \cdot S'_k$ is similar. In Observation 4.2 we show that T'_{k-1} and S'_k are related to T_{k-1} and T_k respectively for $k \in [1, d-2]$.

▶ **Observation 4.2** (Structure of T'_{k-1} and S'_k)**.** *The matrices T'_{k-1} and S'_k computed at Step 5 of Algorithm 2, where $k \in [1, d-2]$, satisfy the following: $(T'_{k-1})^{-1} = T_{k-1}^{-1} \cdot (I_w \otimes C_k) \cdot (M_k^{-1} \otimes I_w)$ and $S'_k = (M_k \otimes I_w) \cdot (I_w \otimes D_k) \cdot T_k$, where $M_k \in GL(w, \mathbb{F})$.*

Steps 6–8: Observation 4.3 describes the structure of the matrices $\widehat{Y}_0, \ldots, \widehat{Y}_{d-1}$ computed at Step 6. Clearly, $\widehat{Y}_0 \ldots \widehat{Y}_{d-1} = Y'_0 \ldots Y'_{d-1}$ is a set-multilinear ABP computing h.

▶ **Observation 4.3.** *Let $M_1, \ldots, M_{d-2}$ be the matrices as defined in Observation 4.2. Then*
1. $\widehat{Y}_k = (M_k M_{k+1}^{-1} \otimes I_w) \cdot (I_w \otimes (C_k^{-1} \cdot X_k \cdot C_{k+1}))$ *for* $k \in [1, d-3]$,
2. $\widehat{Y}_{d-2} = I_w \otimes (C_{d-2}^{-1} \cdot X_{d-2} \cdot D_{d-2}^{-1})$,
3. $\widehat{Y}_0 = Y_0 \cdot (I_w \otimes C_1) \cdot (M_1^{-1} \otimes I_w)$, *and* $\widehat{Y}_{d-1} = (M_{d-2} \otimes I_w) \cdot (I_w \otimes D_{d-2}) \cdot Y_{d-1}$.

By the above observation, at Step 7, $\widehat{X}_{d-2} = C_{d-2}^{-1} \cdot X_{d-2} \cdot D_{d-2}^{-1}$. Moreover, the structure of $\widehat{Y}_k$ (as stated in the observation) enables the algorithm to factor it in Step 7 and obtain $\widehat{X}_k, \widehat{M}_k$ such that

$$\widehat{X}_k = a_k(C_k^{-1} \cdot X_k \cdot C_{k+1}) \quad \text{and } \widehat{M}_k = a_k^{-1}(M_k \cdot M_{k+1}^{-1}) \qquad \text{for some } a_k \in \mathbb{F}^\times.$$

Let $a = \prod_{k=1}^{d-3} a_k$. Then at step 8, $\overline{Y}_{d-1} = a^{-1} \cdot (M_1 \otimes I_w) \cdot (I_w \otimes D_{d-2}) \cdot Y_{d-1}$. Now, it is a simple exercise to verify that at step 8

$$\widehat{X}_0 = (M_1^T)^{-1} \cdot X_0 \cdot C_1 \quad \text{and} \quad \widehat{X}_{d-1} = a^{-1}(D_{d-2} \cdot X_{d-1} \cdot M_1^T).$$

Step 9: Therefore, $h = \mathrm{tr}(\widehat{X}_0 \ldots \widehat{X}_{d-1})$. The transformation $B_k \in \mathrm{GL}(w^2, \mathbb{F})$ is such that its rows are the coefficient vectors of the linear forms in $\widehat{X}_k$.
Hence, $h = \text{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$.

5 Reduction from FMAI to MMTI: Proof of Theorem 2

5.1 Characterization of Tr-IMM by its Lie algebra

The following lemma gives a characterization of $\text{Tr-IMM}_{w,d}$ by its Lie algebra. The spaces $\mathcal{B}_0, \ldots, \mathcal{B}_{d-1}$ are as defined in Section 3.

▶ **Lemma 17.** *Let f be a non-zero d-tensor in the variable sets $\mathbf{x}_0, \ldots, \mathbf{x}_{d-1}$ such that for all $k \in [0, d-1]$ $\mathcal{B}_k \subseteq \mathfrak{g}_f$. Then there is an $\alpha \in \mathbb{F}^\times$ such that $f(\mathbf{x}) = \alpha \cdot \textit{Tr-IMM}_{w,d}(\mathbf{x})$.*

▶ **Corollary 18.** *Let $B \in GL(n, \mathbb{F})$ be a block-diagonal matrix with individual blocks $B_0, \ldots, B_{d-1}$ and f be a non-zero d-tensor in the variable sets $\mathbf{x}_0, \ldots, \mathbf{x}_{d-1}$ such that for all $k \in [0, d-1]$, $B^{-1} \cdot \mathcal{B}_k \cdot B \subseteq \mathfrak{g}_f$. Then there is an $\alpha \in \mathbb{F}^\times$ such that $f(\mathbf{x}) = \alpha \cdot \textit{Tr-IMM}_{w,d}(B_0\mathbf{x}_0, \ldots, B_{d-1}\mathbf{x}_{d-1})$.*

5.2 Proof of Theorem 2

Algorithm 3 takes as input a basis $\{E_1, E_2, \ldots, E_r\}$ of an algebra $\mathcal{A} \subseteq \mathcal{M}_m(\mathbb{F})$, and if $\mathcal{A} \cong \mathcal{M}_w$ for some $w \in \mathbb{N}$, then it computes a 4-tensor f in the variable sets $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ in deterministic polynomial time such that f is isomorphic to $\text{Tr-IMM}_{w,4}$. It then uses the algorithm in Theorem 4 to find an isomorphism from f to $\text{Tr-IMM}_{w,4}$ using oracle access to MMTI in randomized polynomial time. An easy check at the end of the algorithm ensures that if the algorithm outputs an isomorphism then it is correct. Thus, we need to prove that if $\mathcal{A}$ is isomorphic to $\mathcal{M}_w$ for some $w \in \mathbb{N}$ then the algorithm outputs an isomorphism. This is argued by tracing the steps of the algorithm assuming $\mathcal{A}$ is isomorphic to $\mathcal{M}_w$ for some $w \in \mathbb{N}$.

Steps 1–2: At Step 2 there is a $K \in \text{GL}(w^2, \mathbb{F})$ and a basis $\{C_{1,1}, \ldots, C_{w,w}\}$ of $\mathcal{M}_w$ such that $L_{i,j} = K^{-1} \cdot (I_w \otimes C_{i,j}) \cdot K$ for all $i, j \in [w]$ (by the Skolem-Noether theorem, see next claim).

▷ Claim 19. Suppose $\mathcal{A} \cong \mathcal{M}_w$ for some $w \in \mathbb{N}$. Then there exists a $K \in \text{GL}(w^2, \mathbb{F})$ and linearly independent matrices $\{C_{1,1}, \ldots, C_{w,w}\}$ in $\mathcal{M}_w$ such that $L_{i,j} = K^{-1} \cdot (I_w \otimes C_{i,j}) \cdot K$ for all $i, j \in [w]$.

Step 3: The space spanned by $\{L^T_{1,1}, \ldots, L^T_{w,w}\}$ is $K^T \cdot (I_w \otimes \mathcal{M}_w) \cdot (K^T)^{-1}$.

▶ **Observation 5.1.** *The space of matrices in $\mathcal{M}_{w^2}$ that commute with every matrix in $K^T \cdot (I_w \otimes \mathcal{M}_w) \cdot (K^T)^{-1}$ is $K^T \cdot (\mathcal{M}_w \otimes I_w) \cdot (K^T)^{-1}$. So, $\{N_{1,1}, \ldots, N_{w,w}\}$ is a basis of $K^T \cdot (\mathcal{M}_w \otimes I_w) \cdot (K^T)^{-1}$.*

Step 4: Let $n = 4w^2$. For $k \in [0, 3]$, let $\mathcal{B}'_k$ be the following spaces: Every matrix in $\mathcal{B}'_k$ is a $n \times n$ block-diagonal matrix (with rows and columns indexed by $\mathbf{x}_0, \ldots, \mathbf{x}_3$) and its non-zero entries are confined to the rows and columns indexed by $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. For $B \in \mathcal{B}_k$, let $[B]_k$ be the $2w^2 \times 2w^2$ sub-matrix of B as defined in Equation 1 (Section 3). Then

$$\mathcal{B}'_k := \left\{ B \in \mathcal{M}_n \;:\; [B]_k = \begin{bmatrix} K^T \cdot (I_w \otimes M^T)(K^T)^{-1} & \mathbf{0} \\ \mathbf{0} & K^{-1} \cdot (-I_w \otimes M) \cdot K \end{bmatrix} \text{ for } M \in \mathcal{M}_w \right\} \quad \text{if } k \text{ is even,}$$

$$:= \left\{ B \in \mathcal{M}_n \;:\; [B]_k = \begin{bmatrix} K^{-1} \cdot (M^T \otimes I_w) \cdot K & \mathbf{0} \\ \mathbf{0} & K^T \cdot (-M \otimes I_w) \cdot (K^T)^{-1} \end{bmatrix} \text{ for } M \in \mathcal{M}_w \right\} \quad \text{if } k \text{ is odd.}$$

The following observation follows from Lemma 9 and Fact 1.

Algorithm 3 Reduction from FMAI to MMTI.

INPUT: A basis $\{E_1, E_2, \ldots, E_r\}$ of an algebra $\mathcal{A} \subseteq \mathcal{M}_m(\mathbb{F})$, and oracle access to MMTI.
OUTPUT: If $\mathcal{A} \cong \mathcal{M}_w(\mathbb{F})$ for some $w \in \mathbb{N}$ then output an algebra isomorphism $\phi : \mathcal{A} \to \mathcal{M}_w$, otherwise output "No $w \in \mathbb{N}$ such that $\mathcal{A} \cong \mathcal{M}_w$".

1: If $r \neq w^2$ for any $w \in \mathbb{N}$, then output "No $w \in \mathbb{N}$ such that $\mathcal{A} \cong \mathcal{M}_w$".

2: Rename and order the basis elements as $E_{1,1}, \ldots, E_{1,w}, \ldots, E_{w,1}, \ldots, E_{w,w}$. Compute matrices $L_{1,1}, \ldots, L_{w,w}$, whose rows and columns are indexed by the above basis elements in order, as follows: $L_{i,j}$ is the matrix corresponding to the left multiplication of $E_{i,j}$ on $E_{1,1}, \ldots E_{w,w}$. In particular, $E_{i,j} \cdot E_{i_2,j_2} = \sum_{i_1,j_1 \in [w]} L_{i,j}((i_1,j_1),(i_2,j_2))E_{i_1,j_1}$.

3: Compute a basis of the space spanned by matrices in $\mathcal{M}_{w^2}$ that commute with $\{L^T_{1,1}, \ldots, L^T_{w,w}\}$. If the dimension of this space is not w^2, then output 'No $w \in \mathbb{N}$ such that $\mathcal{A} \cong \mathcal{M}_w$'. Otherwise, let the computed basis be $\{N_{1,1}, \ldots, N_{w,w}\}$.

4: Compute a non-zero 4-tensor f in $\mathbf{x}_0, \ldots, \mathbf{x}_3$ variables whose coefficients satisfy the following equations: a) for all $k \in [0,3]$, k even, and for all $L \in \{L_{1,1}, \ldots L_{w,w}\}$

$$\sum_{i_1,j_1,i_2,j_2 \in [w^2]} L^T((i_1,j_1)(i_2,j_2))x^{(k)}_{i_2,j_2} \frac{\partial f}{x^{(k)}_{i_1,j_1}} - \sum_{i_1,j_1,i_2,j_2 \in [w^2]} L((i_1,j_1)(i_2,j_2))x^{(k+1)}_{j_2,i_2} \frac{\partial f}{x^{(k+1)}_{j_1,i_1}} = 0. \tag{2}$$

b) for all $k \in [0,3]$, k odd, and for all $N \in \{N_{1,1}, \ldots N_{w,w}\}$

$$\sum_{i_1,j_1,i_2,j_2 \in [w^2]} N^T((i_1,j_1)(i_2,j_2))x^{(k)}_{j_2,i_2} \frac{\partial f}{x^{(k)}_{j_1,i_1}} - \sum_{i_1,j_1,i_2,j_2 \in [w^2]} N((i_1,j_1)(i_2,j_2))x^{(k+1)}_{i_2,j_2} \frac{\partial f}{x^{(k+1)}_{i_1,j_1}} = 0. \tag{3}$$

5: Use the algorithm in Theorem 4 on input f and with oracle access to MMTI. If the algorithm outputs 'No' then output 'No $w \in \mathbb{N}$ such that $\mathcal{A} \cong \mathcal{M}_w$'. Otherwise, let B_0, B_1, B_2, B_3 be the output of the algorithm such that $f = \text{Tr-IMM}_{w,4}(B_0\mathbf{x}_0, B_1\mathbf{x}_1, B_2\mathbf{x}_2, B_3\mathbf{x}_3)$.

6: Check if there exist matrices $F_{1,1}, \ldots, F_{w,w} \in \mathcal{M}_w$ such that $B_0 \cdot L^T_{i,j} \cdot B_0^{-1} = I_w \otimes F^T_{i,j}$ and $B_1 \cdot L_{i,j} \cdot B_1^{-1} = I_w \otimes F_{i,j}$ for all $i,j \in [w]$. If such matrices do not exist then output 'No $w \in \mathbb{N}$ such that $\mathcal{A} \cong \mathcal{M}_w$', otherwise output $\phi : \mathcal{A} \to \mathcal{M}_w$, where $\phi(E_{i,j}) = F_{i,j}$ for all $i,j \in [w]$ (extended linearly to the whole of $\mathcal{A}$) as the algebra isomorphism from $\mathcal{A}$ to $\mathcal{M}_w$.

▶ **Observation 5.2.** *The Lie algebra of Tr-IMM*$_{w,4}((K^T)^{-1}\mathbf{x}_0, K\mathbf{x}_1, (K^T)^{-1}\mathbf{x}_2, K\mathbf{x}_3)$ *contains* $\mathcal{B}'_0, \mathcal{B}'_1, \mathcal{B}'_2, \mathcal{B}'_3$.

At Step 4, Algorithm 3 computes a non-zero 4-tensor f such that $\mathcal{B}'_k \subseteq \mathfrak{g}_f$ for all $k \in [0,3]$. Equation 2 ensures $\mathcal{B}'_0$, $\mathcal{B}'_2 \in \mathfrak{g}_f$, and Equation 3 ensures $\mathcal{B}'_1, \mathcal{B}'_3 \in \mathfrak{g}_f$. That the algorithm is able to compute a non-zero f (by solving a linear system) follows from Observation 5.2. Since the number of monomials in f is at most w^8, this step runs in polynomial time.

Step 5: From Corollary 18 it follows that $f(\mathbf{x}) = \alpha \cdot \text{Tr-IMM}_{w,4}((K^T)^{-1}\mathbf{x}_0,\ K\mathbf{x}_1,\ (K^T)^{-1}\mathbf{x}_2, K\mathbf{x}_3)$ for some $\alpha \in \mathbb{F}^{\times}$. Hence, at step 5 with high probability the algorithm in Theorem 4 outputs four matrices $B_0, B_1, B_2, B_3 \in \text{GL}(w^2, \mathbb{F})$ such that $f(\mathbf{x}) = \text{Tr-IMM}_{w,4}\ (B_0\mathbf{x}_0,\ B_1\mathbf{x}_1, B_2\mathbf{x}_2, B_3\mathbf{x}_3)$.

Step 6: Let B be the block-diagonal matrix whose k-th block is B_k, for $k \in [0, 3]$. Since $\mathcal{B}'_0 \subseteq \mathfrak{g}_f$ and $\mathfrak{g}_f = B^{-1} \cdot \mathfrak{g}_{\text{Tr-IMM}} \cdot B$ (from Fact 1), $B \cdot \mathcal{B}'_0 \cdot B^{-1} \subseteq \mathfrak{g}_{\text{Tr-IMM}}$. Observe that every matrix in $B \cdot \mathcal{B}'_0 \cdot B^{-1}$ is block-diagonal with its non-zero entries confined to the first two blocks. Hence, from Lemma 10, and the fact that both the spaces $B \cdot \mathcal{B}'_0 \cdot B^{-1}$ and $\mathcal{B}_0$ have dimension w^2, we have $B \cdot \mathcal{B}'_0 \cdot B^{-1} = \mathcal{B}_0$. In particular, for every $i, j \in [w]$ there is an $F_{i,j} \in \mathcal{M}_w$ such that $B_0 \cdot L_{i,j}^T \cdot B_0^{-1} = I_w \otimes F_{i,j}^T$ and $B_1 \cdot L_{i,j} \cdot B_1^{-1} = I_w \otimes F_{i,j}$. Finally, verify that $\phi(E_{i,j}) = F_{i,j}$ is an algebra isomorphism.

Comparison with [15]: In [15], FMAI is reduced to DET by using the fact that Det_w is characterized by its Lie algebra (see Lemma 7.1 in [15]). If the input algebra $\mathcal{A}$ is isomorphic to $\mathcal{M}_w$ then the algorithm in [15] computes a *degree-w* polynomial f in w^2 variables such that $\mathfrak{g}_f$ contains the Lie algebra of a polynomial equivalent to Det_w. Hence, the time complexity of their algorithm is $w^{O(w)}$. Algorithm 3 follows the same approach, but computes a *degree four* polynomial f such that $\mathfrak{g}_f$ contains the Lie algebra of a polynomial equivalent to $\text{Tr-IMM}_{w,4}$. So, the complexity of this algorithm is $w^{O(1)}$.

References

1. Manindra Agrawal and Nitin Saxena. Automorphisms of finite rings and applications to complexity of problems. In *22nd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2005*, pages 1–17, 2005.
2. Manindra Agrawal and Nitin Saxena. Equivalence of f-algebras and cubic forms. In *23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006*, pages 115–126, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
3. Manuel Araújo. Classification of Quadratic Forms. `https://www.math.tecnico.ulisboa.pt/~ggranja/manuel.pdf`, 2011.
4. László Babai and Lajos Rónyai. Computing irreducible representations of finite groups. *Mathematics of Computation*, 55(192):705–722, 1990.
5. Markus Bläser, Christian Ikenmeyer, Meena Mahajan, Anurag Pandey, and Nitin Saurabh. Algebraic branching programs, border complexity, and tangent spaces. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:31, 2020. URL: `https://eccc.weizmann.ac.il/report/2020/031`.
6. Peter A Brooksbank and James B Wilson. The module isomorphism problem reconsidered. *Journal of Algebra*, 421:541–559, 2015.
7. Peter Bürgisser, Christian Ikenmeyer, and Greta Panova. No occurrence obstructions in geometric complexity theory. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 386–395. IEEE Computer Society, 2016.
8. Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity and beyond. *Foundations and Trends in Theoretical Computer Science*, 6(1-2):1–138, 2011.
9. Suryajith Chillara, Nutan Limaye, and Srikanth Srinivasan. Small-depth multilinear formula lower bounds for iterated matrix multiplication with applications. *SIAM J. Comput.*, 48(1):70–92, 2019.
10. J. E. Cremona, T. A. Fisher, C. O'Neil, D. Simon, and M. Stoll. Explicit n-descent on elliptic curves III. algorithms. *Math. Comput.*, 84(292):895–922, 2015.

11 W. M. Eberly. *Computations for algebras and group representations.* PhD thesis, Department of Computer Science, University of Toronto, 1989.

12 Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth-4 formulas computing iterated matrix multiplication. *SIAM J. Comput.*, 44(5):1173–1201, 2015.

13 Georg Frobenius. Ueber die Darstellung der endlichen Gruppen durch lineare Substitutionen. *Sitzungber. der Berliner Akademie*, 7:994–1015, 1897.

14 Vyacheslav Futorny, Joshua A. Grochow, and Vladimir V. Sergeichuk. Wildness for tensors. *Linear Algebra and its Applications*, 566:212–244, 2019.

15 Ankit Garg, Nikhil Gupta, Neeraj Kayal, and Chandan Saha. Determinant equivalence test over finite fields and over Q. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, pages 62:1–62:15, 2019.

16 Fulvio Gesmundo. Geometric aspects of iterated matrix multiplication. *Journal of Algebra*, 461:42–64, 2016.

17 Fulvio Gesmundo, Christian Ikenmeyer, and Greta Panova. Geometric complexity theory and matrix powering. *Differential Geometry and its Applications*, 55:106–127, 2017.

18 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68. ACM, 1986.

19 Joshua A. Grochow. *Symmetry and equivalence relations in classical and geometric complexity theory.* PhD thesis, The University of Chicago, 2012. Available from `https://www.cs.colorado.edu/~jgrochow/grochow-thesis.pdf`.

20 Joshua A. Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. *CoRR*, abs/1907.00309, 2019.

21 Christian Ikenmeyer and Greta Panova. Rectangular kronecker coefficients and plethysms in geometric complexity theory. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 396–405. IEEE Computer Society, 2016.

22 Gábor Ivanyos, Lajos Rónyai, and Joseph Schicho. Splitting full matrix algebras over algebraic number fields. *Jounral of Algebra*, 354:211–223, 2012.

23 Erich Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990.

24 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the 22nd Symposium on Discrete Algorithms, SODA 2011*, pages 1409–1421, 2011.

25 Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing, STOC 2012*, pages 643–662, 2012. Full text available from `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Projection.pdf`.

26 Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Computational Complexity*, 28(4):749–828, 2019.

27 Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between read-once oblivious algebraic branching programs (roabps) and multilinear depth-three circuits. *ACM Trans. Comput. Theory*, 12(1), 2020. Conference version appeared in the proceedings of STACS 2016.

28 Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of full rank algebraic branching programs. *TOCT*, 11(1):2:1–2:56, 2019. Conference version appeared in the proceedings of CCC 2017.

29 Neeraj Kayal and Chandan Saha. Lower bounds for sums of products of low arity polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:73, 2015. URL: `http://eccc.hpi-web.de/report/2015/073`.

30 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. On the size of homogeneous and of depth-four formulas with low individual degree. *Theory of Computing*, 14(1):1–46, 2018. Conference version appeared in the proceedings of STOC 2016.

31 Adam Klivans and Amir Shpilka. Learning arithmetic circuits via partial derivatives. In *Proceedings of the 16th Conference on Learning Theory, COLT 2003*, pages 463–476, 2003.

32 Mrinal Kumar and Shubhangi Saraf. On the Power of Homogeneous Depth 4 Arithmetic Circuits. *SIAM J. Comput.*, 46(1):336–387, 2017.

33 J. M Landsberg. Geometric complexity theory: an introduction for geometers. *Annali Dell'Universita' Di Ferrara*, 61(1):65–117, 2015.

34 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci.*, 1997, 1997.

35 Ketan Mulmuley and Milind A. Sohoni. Geometric complexity theory I: an approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526, 2001.

36 Ketan Mulmuley and Milind A. Sohoni. Geometric complexity theory II: towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.*, 38(3):1175–1206, 2008.

37 Janaky Murthy, Vineet Nair, and Chandan Saha. Randomized polynomial-time equivalence between determinant and trace-imm equivalence tests. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:91, 2020.

38 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Symposium on Theory of Computing, STOC 1991*, pages 410–418, 1991.

39 Noam Nisan and Avi Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity*, 6(3):217–234, 1997.

40 Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 33–48, 1996.

41 Lajos Rónyai. Simple algebras are difficult. In *Proceedings of the 19th Symposium on Theory of Computing, STOC 1987*, pages 398–408, 1987.

42 Lajos Rónyai. Computing the structure of finite algebras. *J. Symb. Comput.*, 9(3):355–373, 1990.

43 Lajos Rónyai. Algorithmic properties of maximal orders in simple algebras over $\mathbb{Q}$. *Computational Complexity*, 2:225–243, 1992.

44 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. *Github survey*, 2015.

45 Nitin Saxena. *Morphisms of rings and applications to complexity.* PhD thesis, Indian Institute of Technology, Kanpur, 2006.

46 Jean-Pierre Serre. *A Course in Arithmetic.* Springer-Verlag New York, 1973.

47 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

48 Thomas Thierauf. The isomorphism problem for read-once branching programs and arithmetic circuits. *Chicago J. Theor. Comput. Sci.*, 1998, 1998.

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

50 Lars Ambrosius Wallenborn. Computing the hilbert symbol, quadratic form equivalence and integer factoring. Diploma thesis, University of Bonn, 2013.