Electronic Colloquium on Computational Complexity, Report No. 73 (2024)

Trading Determinism for Noncommutativity in Edmonds' Problem

V. Arvind* Abhranil Chatterjee† Partha Mukhopadhyay‡

Abstract

Let $X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_k$ be a partitioned set of variables such that the variables in each part X_i are noncommuting but for any $i \neq j$, the variables $x \in X_i$ commute with the variables $x' \in X_j$. Given as input a square matrix T whose entries are linear forms over $\mathbb{Q}\langle X \rangle$, we consider the problem of checking if T is invertible or not over the universal skew field of fractions of the partially commutative polynomial ring $\mathbb{Q}\langle X \rangle$ [KVV20]. In this paper, we design a deterministic polynomial-time algorithm for this problem for constant k. The special case $k = 1$ is the noncommutative Edmonds' problem (NSingular) which has a deterministic polynomial-time algorithm by recent results [GGdOW16, IQS18, HH21].

En-route, we obtain the first deterministic polynomial-time algorithm for the equivalence testing problem of k-tape *weighted* automata (for constant k) resolving a long-standing open problem [HK91, Wor13]. Algebraically, the equivalence problem reduces to testing whether a partially commutative rational series over the partitioned set X is zero or not [Wor13]. Decidability of this problem was established by Harju and Karhumäki [HK91]. Prior to this work, a *randomized* polynomial-time algorithm for this problem was given by Worrell [Wor13] and, subsequently, a deterministic quasipolynomial-time algorithm was also developed [ACDM21].

*Institute of Mathematical Sciences (HBNI), and Chennai Mathematical Institute, Chennai, India, email: arvind@imsc.res.in.

†Indian Statistical Institute, Kolkata, email: abhneil@gmail.com. Research Supported by INSPIRE Faculty Fellowship provided by the Department of Science and Technology, Government of India.

‡Chennai Mathematical Institute, Chennai, email: partham@cmi.ac.in.

ISSN 1433-8092

Contents

1 Introduction

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n variables and $\mathbb{F}$ be a field. Consider the coefficient matrices $A_0, A_1, \ldots, A_n \in \mathrm{Mat}_s(\mathbb{F})$, and define the $s \times s$ symbolic matrix T as

$$T = A_0 + A_1 x_1 + \ldots + A_n x_n.$$

In 1967, Edmonds introduced the problem of deciding whether T is invertible over the rational function field $\mathbb{F}(x_1, x_2, \ldots, x_n)$ [Edm67], often referred to as the SINGULAR problem. More generally, Edmonds was interested in computing the (commutative) rank of T over the rational function field $\mathbb{F}(x_1, x_2, \ldots, x_n)$. The problem can be restated as computing the maximum rank of a matrix in the affine matrix space generated by the $\mathbb{F}$-linear span of the coefficient matrices $A_i, 1 \leqslant i \leqslant n$. This was further studied by Lovász [Lov89], in the context of graph matching and matroid-related problems. The SINGULAR problem, and more generally the rank computation problem, admits a simple randomized polynomial-time algorithm due to the Polynomial Identity Lemma [Sch80, Zip79, DL78]. However, the quest for an efficient *deterministic* algorithm remains elusive. Eventually, Kabanets and Impagliazzo showed that any efficient deterministic algorithm for SINGULAR will imply a strong circuit lower bound, justifying the elusiveness over the years [KI04]. Interestingly, the rank computation problem admits a deterministic PTAS algorithm [BJP18].

The rank computation problem is also well-studied in the noncommutative setting [Coh95, FR04]. More precisely, T is still a linear matrix but the variables $x_1, x_2, \ldots, x_n$ are noncommuting. The problem of testing whether T is invertible (NSINGULAR), or the rank computation question is naturally addressed over the noncommutative analog of the commutative function field, *the free skew field* $\mathbb{F}\langle\!(X)\!\rangle = \mathbb{F}\langle\!(x_1, x_2, \ldots, x_n)\!\rangle$. The free skew field has been extensively studied in mathematics [Ami66, Ami55, Coh71]. Intuitively, it suffices to state that $\mathbb{F}\langle\!(X)\!\rangle$ is the smallest field over the noncommutative ring $\mathbb{F}\langle X \rangle$.

Two independent breakthrough results showed that NSINGULAR is in P [GGdOW16, IQS18]. The algorithm of Garg, Gurvits, Oliveira, and Wigderson [GGdOW16] is analytic in nature and based on operator scaling which works over $\mathbb{Q}$. The algorithm of Ivanyos, Qiao, and Subrahmanyam [IQS18] is purely algebraic, and it works over $\mathbb{Q}$ as well as fields of positive characteristic. Subsequently, a third algorithm based on convex optimization is also developed by Hamada and Hirai [HH21]. Not only are these beautiful results, but also they have enriched the field of computational invariant theory greatly [BFG+19, DM20].

The main driving motivation for this work is to understand the trade-off between the role of noncommutativity and the complexity of Edmonds' problem. More precisely, let $X_{[k]} = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_k$ be a partitioned set of variables such that the variables in each $X_i : 1 \leqslant i \leqslant k$ are noncommuting and $|X_i| \leqslant n$. However, for each $i \neq j$, the variables in X_i commute with the variables in X_j. Given a linear matrix T with (affine)-linear form entries over $X_{[k]}$, the problem is to decide whether T is invertible or not. Of course, in order to consider the invertibility of T, we need a skew field of the fractions of the partially commutative polynomial ring $\mathbb{F}\langle X_{[k]} \rangle$. A construction of such a skew field (which we call as $\mathfrak{U}_{[k]}$) is known when the characteristic of $\mathbb{F}$ is zero [KVV20, Theorem 1.1]. Given the field $\mathfrak{U}_{[k]}$, the definition of matrix rank is as usual, the maximum size of any invertible submatrix over $\mathfrak{U}_{[k]}$. We define PC-SINGULAR as the problem of checking whether such a linear matrix is invertible over $\mathfrak{U}_{[k]}$ where PC stands for the partially commutative nature of the variables. The main result of this paper is the following theorem.

Theorem 1. *Given an $s \times s$ matrix T whose entries are $\mathbb{Q}$-linear forms over the partially commutative set of variables $X_{[k]}$ (where $|X_i| \leqslant n$ for $1 \leqslant i \leqslant k$), the rank of T over $\mathfrak{U}_{[k]}$ can be computed in deterministic $(ns)^{2^{O(k \log k)}}$ time. The bit complexity of the algorithm is also bounded by $(ns)^{2^{O(k \log k)}}$.*

As a direct corollary of Theorem 1, PC-Singular $\in$ P for $k = O(1)$. Notice that PC-Singular generalizes both NSingular and Singular. For $k = 1$ it is just NSingular and the above theorem implies NSingular is in P. Also, letting $|X_i| = 1$ for each i, it captures the Singular problem with k as a running parameter.

Remark 2. We note two points regarding the choice of the field and the input parameters.

1. Theorem 1 is stated over $\mathbb{Q}$ as the result of [KVV20] works over characteristic zero fields, and we also want that the field arithmetic computation should be efficient. The other ingredients of the proof work also over sufficiently large fields of positive characteristic.

2. For convenience (and w.l.o.g) throughout the paper we assume $s \geqslant n$ and express the run time, bit complexity, and the dimension of the matrices used as a function of s and k only.

It is to be noted that apart from NSingular, the *deterministic polynomial-time* algorithm is known only for a few other special instances of Singular problem defined over linear matrices. We refer the reader to Section 1.2 for more details.

Equivalence testing of multi-tape weighted automata En-route to the proof of Theorem 1, we obtain the first deterministic polynomial-time algorithm for equivalence testing of k-tape weighted automata for $k = O(1)$ resolving a long standing open problem [HK91, Wor13]. Since the equivalence testing problem of multi-tape automata is closely related to the rich domain of trace monoids (or partially commutative monoids), we make a small detour to it.

A trace is a set of strings over an alphabet where certain letters (variables) are allowed to commute and others are not. Historically, traces were introduced by Cartier and Foata to give a combinatorial proof of MacMahon's master theorem [CF69]. The trace monoid or the partially commutative monoid is a monoid of traces. More formally, it is constructed by giving an independence relation on the set of commuting letters. This induces an equivalence relation and partitions the given trace into equivalences classes. The set of equivalence classes themselves form a monoid which is a quotient monoid. This is also called the trace monoid which is a foundational object in concurrency theory [DM97, Maz95].

For us the alphabet is the partitioned set of variables $X_{[k]} = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_k$. The variables in X_i are noncommuting but the variables in X_i and X_j for $i \neq j$ are mutually commuting. Given two $s \times s$ linear matrices T_1, T_2 over $X_{[k]}$and vectors $u_1, u_2 \in \mathbb{F}^{1\times s}$, $v_1, v_2 \in \mathbb{F}^{s\times 1}$, the problem is to check whether the following infinite series are the same:

$$u_1 \left(\sum_{i \geqslant 0} T_1^i\right) v_1 \stackrel{?}{=} u_2 \left(\sum_{i \geqslant 0} T_2^i\right) v_2.$$

Let $X_{[k]}^*$ denote the set of all monomials (or words) over the variables in $X_{[k]}$. Any monomial $m \in X_{[k]}^*$ can be obtained as some interleaving of monomials $m_i \in X_i^*, 1 \leqslant i \leqslant k$. Conversely, given $m \in X_{[k]}^*$ we can uniquely extract each $m_i \in X_i^*$ by dropping from monomial m the variables in $X_{[k]} \setminus X_i$. Essentially each m_i is the restriction $m|_{X_i^*}$. Hence, two partially commutative monomials $m, m' \in X_{[k]}^*$ are the same if and only $m|_{X_i^*} = m'|_{X_i^*}$ for each $1 \leqslant i \leqslant k$. This defines an equivalence relation $\sim$ over the set of monomials $X_{[k]}^*$. To see a simple example, consider $X_1 = \{x_1, x_2\}$ and $X_2 = \{x_1', x_2'\}$. Then $x_1 x_2' x_2 x_1' \sim x_1 x_2 x_2' x_1'$. This is the algebraic formulation of the well-known k-tape weighted automata equivalence problem. See [Wor13, Section 3] for a detailed discussion. Equivalence testing of k-tape weighted automata was shown to be *decidable* by Harju and Karhumäki [HK91] using the theory of free groups. Indeed, a co-NP upper bound follows from their result as observed by Worrell [Wor13]. Improved complexity upper bounds for this problem remained elusive, until Worrell [Wor13] obtained a *randomized* polynomial-time algorithm for testing the equivalence of k-tape weighted automata for any constant k. Worrell's key insight was to reduce this problem to the polynomial identity testing of algebraic branching programs (ABPs) defined over the partially commutative set of variables $X_{[k]}$ (in other words, the linear forms on the edges of the ABP are in $\mathbb{F}\langle X_{[k]}\rangle$). Essentially, the reduction says that two infinite series are the same if and only if

$$u_1\left(\sum_{i\leqslant s} T_1^i\right)v_1 = u_2\left(\sum_{i\leqslant s} T_2^i\right)v_2.$$

This is obtained by adapting such a result for $k = 1$ case suitably for arbitrary k [Eil74, Corollary 8.3]. This is equivalent to the following identity testing problem:

$$u\left(\sum_{i\leqslant s} T^i\right)v \stackrel{?}{=} 0$$

$$\text{where,}\quad u = \begin{pmatrix} u_1 & u_2 \end{pmatrix},\ v = \begin{pmatrix} v_1 \\ -v_2 \end{pmatrix},\ T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}.$$

Clearly, $u\left(\sum_{i\leqslant s} T^i\right)v$ can be represented as an ABP of width $2s$ and degree s defined over the variable set $X_{[k]}$. Then, Worrell developed a partially commutative analogue of the well-known Amitsur-Levitzki Theorem to solve the identity testing problem in randomized polynomial time [AL50, Wor13]. Building on Worrell's work, in [ACDM21] a *deterministic* quasipolynomial-time algorithm was obtained for any constant k. The key technical idea was a bootstrapping of the quasipolynomial-size hitting set for noncommutative ABPs [FS13] to the partially commutative setting. However, the main open question of [HK91, Wor13] was to design a deterministic *polynomial-time* test that remained elusive. In this paper, we fully resolve the problem by giving the first deterministic polynomial-time algorithm for any constant k.

Theorem 3. *Given an ABP of size s whose edges are labeled by $\mathbb{Q}$-linear forms over the partially commutative set of variables $X_{[k]}$ (where $|X_i| \leqslant n$ for $1 \leqslant i \leqslant k$), there is a deterministic $(ns)^{2^{O(k\log k)}}$ time algorithm to check whether the ABP computes the zero polynomial. As a corollary, the equivalence testing of k-tape weighted automata can be solved in deterministic polynomial time for $k = O(1)$. The bit complexity of the algorithm is also bounded by $(ns)^{2^{O(k\log k)}}$.*

As already mentioned in Remark 2, that for convenience we always take $s \geqslant n$. We provide more background and other results related to the equivalence testing problem of multi-tape weighted automata in Section 1.2.

1.1 Proof Idea

When a ring R is embeddable in a skew field $\mathfrak{F}$, the notions of rank and singularity of matrices over R are easier to work with. It is a remarkable fact that in the noncommutative world, even integral domains, in general, need not be embeddable in a skew field! Cohn's text contains a detailed study of matrix rank over different rings [Coh95]. We also refer the reader to the important paper of Malcev [Mal37]. An $s \times s$ matrix T over such a ring R is invertible if there is a matrix T^{-1} over $\mathfrak{F}$

such that $TT^{-1} = T^{-1}T = I_s$.[1] An $s \times s$ matrix T over this ring R is invertible precisely when its rank is s. Likewise, the rank of an $s \times t$ matrix M over such a ring R is precisely the maximum r such that M has an $r \times r$ invertible submatrix. An example of this setting is the free noncommutative ring $R = \mathbb{F}\langle X\rangle$ which embeds in the free skew field $\mathbb{F}\langle\!\langle X \rangle\!\rangle$.

For $S \subseteq [k]$, let X_S be the set of variables in X_i for $i \in S$. Now, if X is a set of partially commutative variables $X = X_{[k]}$, singularity testing (or more generally the rank computation) of any linear matrix T defined over $X_{[k]}$, the construction of a universal skew field containing $\mathbb{F}\langle X_{[k]}\rangle$ will be required. As already mentioned, such a construction is recently obtained [KVV20, Theorem 1.1] when $\mathbb{F}$ is characteristic zero. We will denote that universal skew field by $\mathfrak{U}_{[k]}$. More generally, for a subset of indices $S \subseteq [k]$, we will denote by $\mathfrak{U}_S$ the universal skew field containing the ring $\mathbb{F}\langle X_S\rangle$. This is the main reason that we state our results over fields of characteristic zero and for efficient computational purpose, we fix it to be $\mathbb{Q}$.

We develop two recursive subroutines PC-PIT and PC-Rank which are the building blocks of our main results. The subroutine PC-PIT takes as input an ABP whose edges are labeled by $\mathbb{Q}$-linear forms over the partially commutative variables $X_{[k]}$ and finds matrix assignments of the form 1 to the variables in $X_1, X_2, \ldots, X_k$ such that the nonzeroness is preserved. For clarity, when the subroutine PC-PIT handles ABPs over a ℓ-partition set, we denote it by PC-PIT$_\ell$. For example, here we are interested in PC-PIT$_k$.

The subroutine PC-Rank takes a linear matrix T over $X_{[k]}$ as input and finds matrix assignments to the variables in X of the form 1 that attains the rank. More precisely, if the rank of T in $\mathfrak{U}_{[k]}$ is r and the dimension of the matrices is d, then the rank of the scalar matrix obtained from T after the substitution is rd. We use PC-Rank$_\ell$ to indicate that the subroutine is applied over a ℓ-partition variable set. In essence, it turns out that the two subroutines PC-PIT$_k$ and PC-Rank$_k$ are interlinked. Indeed, PC-PIT$_k$ makes subroutine calls to PC-Rank$_{k-1}$ and, in turn, PC-Rank$_k$ makes subroutine calls to PC-PIT$_k$.

As a warm-up, we first consider PC-Rank subroutine for $k = 1$ case i.e. the NSingular problem. This algorithm for NSingular, reduces the main algorithmic step (which is the rank increment step) to noncommutative ABP identity testing. It allows us to design a new algorithm for NSingular, presented in Section 3. It turns out that this connection to ABP identity testing can be lifted in the setting of partially commutative case and proved to be a key conceptual component in the proofs of Theorem 1 and Theorem 3. We do not know if other algorithms for NSingular, e.g. the algorithm in [IQS18] which is based on the connection between singularity and the existence of shrunk subspaces [FR04], can be generalized to the partially commutative setting.[2]

A crucial notion that plays an algorithmic role in [IQS18] and in our NSingular algorithm is the blow-up rank [DM17, IQS18]. Let T be a linear matrix in noncommutative variables. Writing $T = A_0 + \sum_{i=1}^{n} A_i x_i$, where $A_0, A_1, \ldots, A_n$ are coefficient matrices, the evaluation of T at a matrix tuple $\underline{M} = (M_1, M_2, \ldots, M_n)$ of dimension d, where each M_i has scalar entries is:

$$T(\underline{M}) = A_0 \otimes I_d + \sum_{i=1}^{n} A_i \otimes M_i.$$

Define $T^{\{d\}} = \{T(\underline{M}) \mid \text{each } M_i \in \mathbb{F}^{d\times d}\}$. Notice that $T^{\{d\}}$ contains $sd \times sd$ matrices. Let rank$(T^{\{d\}})$ be the maximum rank attained by a matrix in $T^{\{d\}}$. The regularity lemma [IQS18, DM17] shows that rank$(T^{\{d\}})$ is always a multiple of d. Moreover, ncrank(T) is the maximum r such that for

[1] The inverse if it exists will be unique and is hence denoted T^{-1}.

[2] Neither do we know if the other approaches for NSingular in [GGdOW16, HH21] are applicable in this setting.

some d $\mathrm{rank}(T^{\{d\}}) = rd$. If for a tuple $\underline{M}$ of dimension d the rank of $T(\underline{M}) \geqslant rd$, we say that $\underline{M}$ is a witness of rank r.

Guided by the above notion of blow-up rank, the algorithm in [IQS18] has two main steps applied iteratively: the rank increment step, and the rounding and blow-up control step. We briefly sketch their algorithm. Given a matrix B in $T^{\{d\}}$ of rank $\geqslant rd$, the rank-increment step searches[3] for a new matrix B' in $T^{\{d'\}}$ (where $d' > d$) of rank $\geqslant rd' + 1$. If no such matrix exists, then $\mathrm{ncrank}(T) = r$ where $\mathrm{ncrank}(T)$ is the rank of T in $\mathbb{F}\langle\!\langle X \rangle\!\rangle$. Next, the rounding step is a constructive version of the regularity lemma to find another matrix B'' in $T^{\{d'\}}$ such that the rank of B'' is $r'd'$ where r' is at least $r + 1$. A blow-up in the dimension of $\underline{M}$ at each iteration incurs an exponential blow-up in the final dimension. They control the dimension increase by dropping rows and columns from the witness matrices along with repeated applications of the rounding step. Finally, it outputs a matrix $\hat{B}$ of rank $r'd''$ where $r' \geqslant r + 1$ and $d'' \leqslant r' + 1$. The rounding step crucially works with matrices from a division algebra (because nonzero matrices in a division algebra are of full rank).

Coming back to our NSingular algorithm, the rounding and blow-up control step is very similar to that in [IQS18]. As already mentioned, the main difference is the rank increment step which we reduce to PIT of noncommutative ABPs. As we show in Lemma 18, Lemma 30, and Corollary 31, given a linear matrix T and a rank-r witness of dimension d, it essentially suffices to compute a nonzero matrix tuple for a noncommutative ABP of size rd to find a witness of T of noncommutative rank $r + 1$. This can be done with well-known identity testing algorithms [RS05, AMS10]. This also avoids incurring any super-polynomial bit-complexity blow-up over $\mathbb{Q}$.

Armed with the intuition for the new algorithm for NSingular, we now sketch the main ideas of the proofs of Theorem 1 and Theorem 3. It is shown in [KVV20] that a linear matrix T over the partially commutative variable set $X_{[k]}$ is invertible (over the universal skew field $\mathfrak{U}_{[k]}$) if and only if there exists matrix substitutions for the variables $x \in X_i : 1 \leqslant i \leqslant k$ of the form

$$I_{d_1} \otimes I_{d_2} \otimes \cdots \otimes I_{d_{i-1}} \otimes M_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k} \tag{1}$$

such that T evaluates to an invertible matrix. Here, M_x is a $d_i \times d_i$ matrix and $d_1, d_2, \ldots, d_k \in \mathbb{N}$. Notice that the structure of the matrices respect the partial commutativity. The basic idea in our proof is to explicitly (and efficiently) find such matrices respecting partially commutative tensor product structures. Our algorithm also confirms that each dimension d_i is at most $s + 1$.[4]

ABP identity testing over partially commutative variables We now give an overview of the PC-PIT$_k$ subroutine. In the noncommutative case ($k = 1$), when X is just a set of noncommuting variables, the PIT algorithm in [RS05], first homogenizes the ABP using standard techniques [RS05, Lemma 2] and then identity tests each homogenized component. Each homogenized ABP is processed layer by layer. An important feature of homogeneous noncommutative ABPs is that every nonzero monomial m has *unique parsing*: more precisely, the only way the ABP can construct a monomial m is from left to right, one variable at a time. This allows the algorithm of [RS05] to maintain at layer i (of width w) at most w monomials of degree i that have linearly independent coefficient vectors at that layer.

This crucial unique parsing property does not hold for ABPs defined over partially commutative variables (for $k > 1$). To handle this, we homogenize the input ABP $\mathcal{A}$ over the variable set X_1,

[3]Computing the limit point of a second Wong sequence [IKQS15, IQS18], a non-trivial generalization of augmenting paths algorithm in the bipartite graph matching.

[4]For $k = 1$, the result in [DM17, Theorem 1.8] shows that for linear matrices of size s, $s - 1$ dimension suffices.

treating the remaining variables as part of the coefficients. More precisely, suppose the input ABP $\mathcal{A}$ is of width w, degree d and size s. Then it turns out each X_1-homogenized component is an ABP whose edge labels are linear forms $\sum_i \alpha_i x_i$, with $x_i \in X_1$, such that the coefficients α_i are given by ABPs of size $O(sd) = O(s^2)$ over variables $X_2, X_3, \ldots, X_k$ (Lemma 12). For an X_1-homogenized ABP, inductively, assume that at the j^{th} level, we have recorded the monomials $m_1, m_2, \ldots, m_w \in X_1^j$ and the corresponding coefficient vectors are $v_1, v_2, \ldots, v_w$. The entries of the vectors v_i are ABPs over $X_2, X_3, \ldots, X_k$ of size $O(s^2 j)$. The vector v_i is the vector of coefficients of the monomial m_i in the polynomials computed at each node of layer j. Additionally, we maintain the property that the vectors $v_1, v_2, \ldots, v_w$ are $\mathfrak{U}_{[k]\setminus\{1\}}$-linearly independent and also $\mathfrak{U}_{[k]\setminus\{1\}}$-spanning for the set of all vectors corresponding to all monomials in X_1^j (spanning as a left $\mathfrak{U}$-module). For the $(j+1)^{th}$ level, we need to now compute a similar set of $\mathfrak{U}_{[k]\setminus\{1\}}$-linearly independent vectors from among the vectors corresponding to the monomials $\{m_i x_j : 1 \leqslant i \leqslant w, x_j \in X_1\}$. Clearly the size of such a set is bounded by the width of the ABP. We will see that this is reducible to computing the rank of a matrix M whose entries are ABPs over the variables in $X_{[k]\setminus\{1\}}$. It turns out that, we can linearize this rank problem by adapting a recent result [ACG+22] proved in the context of noncommutative ($k = 1$) setting. That is, the rank computation of matrix M is polynomial-time reducible to the rank computation of a *linear* matrix T over $X_2, X_3, \ldots, X_k$ of size $O(s^5)$. This is the place where we recursively call PC-Rank$_{k-1}$ for linear matrices of size $O(s^5)$ over the variable set $X_{[k]\setminus\{1\}}$.

At the end of this process we find a surviving monomial m_1 over the variables in X_1 whose coefficient is nonzero. Given such a monomial, we can use a standard idea (by now) to produce assignments $\{M_x\}_{x\in X_1}$ which makes the polynomial evaluate to a nonzero matrix of polynomials over $X_2, X_3, \ldots, X_k$ [AMS10]. The dimension of the matrices $\{M_x\}_{x\in X_1}$ is bounded by $\deg(f) + 1$ (recall that f is the polynomial computed by the ABP), and the entries are over $\{0, 1\}$.

One can then recover the ABP $\mathcal{A}_{m_1}$ over $X_2, X_3, \ldots, X_k$ which is the coefficient of m_1. The size of the ABP will be $O(sd^2) = O(s^3)$, however the degree is still bounded by $\deg(f)$. It now recursively invokes PC-PIT$_{k-1}$ over $\mathcal{A}_{m_1}$ to compute the matrix assignments for the variables in $X_{[k]\setminus\{1\}}$ such that $\mathcal{A}_{m_1}$ evaluates to a nonzero matrix.

Since the dimension of the matrices is only a function of $\deg(f)$, it is always bounded by $s + 1$ for each X_i. Finally, the matrix substitution for the variables $x \in X_i$ will be identified with the form 1. To summarize, the upshot is that the PC-PIT$_k$ problem over $X_{[k]}$ is deterministic poly(s)-time reducible to $O(s^4)$ instances of PC-Rank$_{k-1}$ for linear matrices over $X_2, X_3, \ldots, X_k$ of size $O(s^5)$ and at most s recursive calls to PC-PIT$_{k-1}$ for ABPs of size $O(s^3)$ defined over $X_{[k]\setminus\{1\}}$ (taking all homogenized components into account).

Computing linear matrix rank over partially commutative variables Now we discuss the construction of the subroutine PC-Rank$_k$. Given a linear matrix T of size s over the partially commutative variables $X_{[k]}$, let pc-rank(T) denote its rank over the universal skew field $\mathfrak{U}_{[k]}$. This is the size of the largest invertible submatrix (over $\mathfrak{U}_{[k]}$) of T. The subroutine PC-Rank$_k$ finds the matrix assignments of the form 1 to the variables such the rank of the new scalar matrix becomes a multiple of pc-rank(T). More precisely, the rank of the scalar matrix after the matrix assignments is $d' \cdot$ pc-rank(T) where $d' = d_1 d_2 \cdots d_k$.

Let us define the notion of the witness for pc-rank. A set of matrix tuples of the form 1 is a rank r witness for T if after the substitution, the rank of the scalar matrix is at least rd'. Now to construct the subroutine PC-Rank$_k$, the main idea is to do an induction over r. Namely, given a witness for pc-rank r (which we call as the matrix tuple $\underline{M}$), we would like to construct another witness for pc-rank $r + 1$ in deterministic polynomial time unless r is already the pc-rank(T). Note that to construct a witness for rank $r = 1$, it suffices to assign values to the variables such that any nonzero linear form in T becomes nonzero, which is clearly trivial.

Let the input matrix T (of size s) be of the following form:

$$T(X_1, \ldots, X_k) = A_0 + \sum_{j=1}^{k} \sum_{x \in X_j} A_x x.$$

Given a pc-rank witness r for T of the form 1, the rank of

$$T_1'' = A_0 \otimes I_{d'} + \sum_{j=1}^{k} \sum_{x \in X_j} A_x \otimes (I_{d_1} \otimes \cdots I_{d_{j-1}} \otimes M_x \otimes I_{d_{j+1}} \otimes \cdots \otimes I_{d_k}) \tag{2}$$

is at least rd' where T_1'' is the evaluation of T on the witness tuple. Additionally, assume that for $1 \leqslant j \leqslant k$, the dimension $d_j \leqslant s^3$. We call $\underline{d} = (d_1, d_2, \ldots, d_k)$ as the shape of the tensor product.

Let $T_{d'}(Z)$ denote the matrix obtained from T by replacing the variable $x \in X_i$ by the matrix $I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$ where the dimension of the generic matrix Z_x is d_i.

By generic, we mean that the entries of Z_x are indeterminate variables $z_{x,\ell_1,\ell_2} : 1 \leqslant \ell_1, \ell_2 \leqslant d_i$. Furthermore, the variables in $Z_i = \{Z_x\}_{x \in X_i}$ are noncommuting but variables across Z_i and Z_j are commuting for $i \neq j$. Equivalently, one can view each Z_i as the set of variables $\{z_{x,\ell_1,\ell_2}\}_{x \in X_i, 1 \leqslant \ell_1, \ell_2 \leqslant d_i}$. Thus we have a new set of partially commutative variables over $Z = (Z_1, \ldots, Z_k)$ but with equal number of partitions. It is important to note that pc-rank$(T_{d'}(Z)) = d' \cdot$ pc-rank(T) (Corollary 41). We require a similar observation for our NSingular algorithm (Lemma 28).

In $T_{d'}(Z)$ replace the matrices $I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$ corresponding to $x \in X_i$ by

$$I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes (Z_x + M_x) \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$$

and obtain the matrix $T_{d'}(Z + \underline{M})$. Note that the scalar part of the matrix is T_1''. In other words,

$$T_{d'}(Z + \underline{M}) = T_1'' + \sum_{i=1}^{k} \sum_{x \in X_i} A_x \otimes I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}. \tag{3}$$

By the simple property that the rank of a linear matrix is invariant under shifting of the variables by scalars, we get that pc-rank$(T_{d'}(Z + \underline{M}))$ = pc-rank$(T_{d'}(Z))$.

Applying Gaussian elimination, we can transform $T_{d'}(Z + \underline{M})$ to the following shape:

$$T_{d'}(Z + \underline{M}) \to \left(\begin{array}{c|c} I_{rd'} - L & 0 \\ \hline 0 & C - B(I_{rd'} - L)^{-1} A \end{array} \right). \tag{4}$$

The matrices L, A, B, C are linear matrices over the variables in $Z_1, Z_2, \ldots, Z_k$. The $(\ell_1, \ell_2)^{th}$ entry of $C - B(I_{rd'} - L)^{-1} A$ is given by $S_{\ell_1,\ell_2} = C_{\ell_1 \ell_2} - B_{\ell_1}(I_{rd'} - L)^{-1} A_{\ell_2}$ where B_{ℓ_1} is the ℓ_1^{th} row vector of B and A_{ℓ_2} is the ℓ_2^{th} column vector of A. Now we notice a simple fact that shows pc-rank$(T) > r$ if and only if $S_{\ell_1,\ell_2} \neq 0$ for at least one pair (ℓ_1, ℓ_2) (Lemma 42). This is the partially commutative version of Lemma 29 that we prove in the context of NSingular problem.

Notice that S_{ℓ_1,ℓ_2} has the following series expansion

$$S_{\ell_1 \ell_2} = C_{\ell_1 \ell_2} - B_{\ell_1} \left(\sum_{i \geqslant 0} L^i \right) A_{\ell_2}.$$

The refined goal is to find a nonzero for the series which allows us to construct a witness of pc-rank$(T) \geqslant r+1$. If the series is defined over the free noncommuting variables, a standard result [Eil74, Corollary 8.3] shows that the infinite series is nonzero if and only if the polynomial

$$S^{\leqslant rd'}_{\ell_1\ell_2} = C_{\ell_1\ell_2} - B_{\ell_1}\left(\sum_{i\leqslant rd'} L^i\right)A_{\ell_2} \neq 0.$$

The same result can be extended to the partially commutative case to obtain a similar statement. In [Wor13, Proposition 5], Worrell proves this statement using Ore domains. A self contained proof is given in Lemma 18. The important (and simple) observation is that the polynomial $S^{\leqslant rd'}_{\ell_1\ell_2}$ can be represented by a partially commutative ABP of width $\leqslant rd'$ and degree $rd'+2$ over the variable set $Z_1, Z_2, \ldots, Z_k$.

Hence, we can apply PC-PIT$_k$ subroutine on the ABP computing the partially commutative polynomial $S^{\leqslant rd'}_{\ell_1\ell_2}$. Additionally, we observe that a suitable scaling of the nonzero of the ABP will be a nonzero for the infinite series $S_{\ell_1\ell_2}$ also. This is by the combined effect of applying Theorem 36 and Lemma 38. As a result, we obtain a matrix tuple that witness the pc-rank$(T) > r$. Now we need a rounding operation that should produce a witness for pc-rank$(T) \geqslant r+1$ and also a blow-up control procedure that controls the dimension of the matrices. This step is somewhat similar in spirit to the rounding and blow-up control steps for the NSingular algorithm, but requires additional conceptual ideas. More precisely, given a linear matrix T over $X_{[k]}$ of size s and matrix tuple of shape $(d_1, \ldots, d_k)$ such that the rank of the image $> rd_1d_2\cdots d_k$, our idea is to update the matrix substitution such that the rank of the image of the new substitution $\geqslant (r+1)d_1d_2\cdots d_k$. Indeed, assuming that the d_i are pairwise relatively prime, the rounding step turns out to be essentially like the noncommutative case. However, this assumption makes the blow-up control step harder. Even if we start with a substitution of shape $(d_1, \ldots, d_k)$ where each d_i is prime, it might fail for $d_i - 1$. To overcome this, our idea is to relax the dimension upper bound of the witness matrix. Instead of reducing d_i one at a time, we allow it to drop to the next (suitable) prime number less than d_i. A theorem about the distribution of primes in small intervals helps us find such a prime close enough to d_i [LS12].

1.2 Other Related Results

Among the specific instances of Singular problem, a deterministic polynomial-time algorithm is known if the coefficient matrices of the symbolic matrix is of rank-one or rank-two skew-symmetric [Lov89]. Raz and Wigderson have given a deterministic polynomial-time algorithm for another instance of Singular problem originated in the context of graph rigidity [RW19]. Another result by Ivanyos, and Qiao gives deterministic polynomial-time algorithm for a special case of Singular problem related to symmetrization or skew-symmetrization problem [IQ19]. Recently, Ivanyos, Mittal, and Qiao obtain a deterministic polynomial-time algorithm where the coefficient matrices generate a matrix Lie algebra [IMQ22].

Equivalence testing of multitape automata is a foundational algorithmic question and has a long history. One-way multitape automata were introduced in the seminal paper of Rabin and Scott [RS59]. The equivalence testing problem of multitape nondeterministic automata is undecidable [Gri68]. Here the equivalence means the words accepted as sets and the question is to decide whether two sets are the same. The problem was shown to be decidable for 2-tape *deterministic* automata independently by Bird [Bir73] and Valiant [Val74]. Subsequently, an exponential upper bound was obtained for it [Bee76]. Eventually, for two-tape deterministic automata, a polynomial-time algorithm was given in [FG82]. As already mentioned, using the theory of free groups, Harju and Karhumäki [HK91] established the decidability of *multiplicity equivalence* of multitape nondeterministic automata. More generally, they prove that the weighted equivalence testing of multitape automata is decidable. One of their open questions was to give an efficient algorithm for the weighted equivalence testing problem when the number of tapes is any constant. Worrell's result giving a randomized polynomial-time algorithm is the first major progress in this direction [Wor13], followed by the quasipolynomial deterministic bound given in [ACDM21]. A relatively recent result analyzes the combinatorial method of Bird [Bir73] more carefully, and it shows a polynomial-time algorithm for the equivalence problem for k-tape *deterministic* automata (where the coefficients are only $0-1$) when $k = O(1)$ [GS20]. Our paper completes this line of investigation by obtaining the first deterministic polynomial-time equivalence test for weighted k-tape automata for $k = O(1)$, thereby improving on the previous algorithmic results.

Organization.

In Section 2, we collect background results from algebraic complexity theory and cyclic division algebras. We give the algorithm for noncommutative singularity testing in Section 3. The main results (Theorem 1 and Theorem 3) are proved in Section 4. We state a few question for further research in Section 5.

2 Background and Notation

Throughout the paper, we use $\mathbb{F}, F, K$ to denote fields. $\mathrm{Mat}_m(\mathbb{F})$ (or $\mathrm{Mat}_m(F), \mathrm{Mat}_m(K)$) will denote m-dimensional matrix algebras over $\mathbb{F}$ (resp. F or K) where m will be clear from the context. Similarly, $\mathrm{Mat}_m(\mathbb{F})^n$ (resp. $\mathrm{Mat}_m(F)^n, \mathrm{Mat}_m(K)^n$) will denote the set of n tuples over $\mathrm{Mat}_m(\mathbb{F})$ (resp. $\mathrm{Mat}_m(F), \mathrm{Mat}_m(K)$). D is used to denote a division algebra. We use X to denote a set of variables. Sometimes, we use $\underline{p}, \underline{q}, \underline{M}, \underline{N}$ to denote matrix tuples in suitable matrix algebras. The free noncommutative ring or partially commutative ring of polynomials over a field $\mathbb{F}$ is denoted by $\mathbb{F}\langle X\rangle$ where X is clear from the context. The notation $A \otimes B$ denotes the usual tensor product of the matrices A and B. We use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. Let $X = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_k$. For $S \subseteq [k]$, let X_S be the set of variables in $\bigsqcup_{i\in S} X_i$. In particular, if X is a set of partially commutative variables, it is denoted by $X_{[k]}$.

2.1 Algebraic Complexity

Definition 4 (Algebraic Branching Program)**.** An *algebraic branching program* (ABP) is a layered directed acyclic graph. The vertex set is partitioned into layers $0, 1, \ldots, d$, with directed edges only between adjacent layers (i to $i + 1$). There is a *source* vertex of in-degree 0 in the layer 0, and one out-degree 0 *sink* vertex in layer d. Each edge is labeled by an affine $\mathbb{F}$-linear form. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the ordered product of affine forms labeling the path edges.

The *size* of the ABP is defined as the total number of nodes and the *width* is the maximum number of nodes in a layer. The ABP model can compute commutative or noncommutative polynomials (depending on the variable set X). ABPs of width w can also be seen as iterated matrix multiplication $\underline{c} \cdot M_1 M_2 \cdots M_\ell \cdot \underline{b}$, where $\underline{c}, \underline{b}$ are $1 \times w$ and $w \times 1$ vectors respectively and each M_i is a $w \times w$ matrix, whose entries are affine linear forms over $\underline{x}$.

Similarly, the ABP model can be used to compute polynomials over a set $X_{[k]}$ of partially commutative variables. The only difference is that the linear forms are over $\mathbb{F}\langle X_{[k]}\rangle$ and two monomials $m, m' \in X^*$ are same under the equivalence relation $\sim$ as described in Section 1.

Definition 5 (Linear Pencil for Noncommutative Polynomials)**.** A noncommutative polynomial $g \in \mathbb{F}\langle X\rangle$ is said to have a size s *linear pencil* L if L is an $s \times s$ invertible linear matrix over X such that g is computed in the $(1, s)^{th}$ entry of L^{-1}.

We can now generalize Definition 5 for partially commutative polynomials also where $X = X_{[k]}$.

Definition 6 (Linear Pencil for Partially Commutative Polynomials)**.** A partially commutative polynomial $g \in \mathbb{Q}\langle X_{[k]}\rangle$ is said to have a size s *linear pencil* L if L is an $s \times s$ invertible linear matrix over $X_{[k]}$ such that g is computed in the $(1, s)^{th}$ entry of L^{-1}.

Since we will be using the result in [KVV20] throughout the paper, whenever we talk about invertibility over the partially commutative setting, the field is always fixed to be $\mathbb{Q}$.

Given an ABP that computes a polynomial f at the the $(1, w)^{th}$ entry of the matrix product $M_1 M_2 \cdots M_d$ where each M_i is of size $w \times w$, it is well-known that the polynomial can be computed at the upper right corner of the inverse of a linear matrix L_f of small size. This was explicitly stated in [HW15, Equation 6.4] in the context of noncommutative variables. However, we can immediately see that the construction also holds for a partially commutative set of variables. We give the formal statement.

Proposition 7. *An ABP of size s (width w, and depth d) computing a polynomial f over the partially commutative variables $X_{[k]} = \bigsqcup_{i=1}^{k} X_i$ has the following linear pencil of size bounded by $2s$:*

$$L_f = \begin{bmatrix} I_w & -M_1 & & & \\ & I_w & -M_2 & & \\ & & \ddots & \ddots & \\ & & & I_w & -M_d \\ & & & & I_w \end{bmatrix}.$$

The polynomial f is computed at the upper right corner of L_f^{-1}.

We also record the following simple observation that talks about the partial evaluation of a polynomial defined over $X_{[k]}$. This is field independent.

Observation 8. *Let $f \in \mathbb{F}\langle X_{[k]}\rangle$ be a partially commutative polynomial. For each $i \in [k]$ and $x \in X_i$, let M_x be a $d_i \times d_i$ matrix. Consider the following matrices:*

1. *Substitute each $x \in X_1$ by M_x in f and obtain a $d_1 \times d_1$ matrix $M_1 \in \mathrm{Mat}_{d_1}(\mathbb{F}\langle X_{[k]\setminus\{1\}}\rangle)$. Similarly, define a $(d_1 d_2 \cdots d_i) \times (d_1 d_2 \cdots d_i)$ matrix $M_i \in \mathrm{Mat}_{d_1 d_2 \cdots d_i}(\mathbb{F}\langle X_{[k]\setminus[i]}\rangle)$ by substituting each $x \in X_i$ by M_x in the $(d_1 d_2 \cdots d_{i-1}) \times (d_1 d_2 \cdots d_{i-1})$ matrix $M_{i-1} \in \mathrm{Mat}_{d_1 d_2 \cdots d_{i-1}}(\mathbb{F}\langle X_{[k]\setminus[i-1]}\rangle)$. Let M_k be the final matrix.*
2. *Let M^* be the matrix evaluation of $f(X_{[k]})$ substituting for each $i \in [k]$, each $x \in X_i$ by*

$$I_{d_1} \otimes \cdots I_{d_{i-1}} \otimes M_x \otimes I_{d_{i+1}} \cdots I_{d_k}.$$

Then, it computes the same matrix i.e. $M_k = M^*$.

Proof. Consider a monomial m in $f(X_{[k]})$. We can write $m = m_1 m_2 \cdots m_k$ where $m_i \in X_i^*$. Let $N_{i,m} = \prod_{x \in m_i} M_x$ be the $d_i \times d_i$ matrix. Now, $M_1 = \sum_m N_{1,m} \otimes m_2 \otimes \cdots \otimes m_k$ from the definition. Therefore, $M_k = \sum_m N_{1,m} \otimes N_{2,m} \otimes \cdots \otimes N_{k,m}$. Clearly from the definition of each $N_{i,m}$ the contribution of each m in M^* is also $N_{1,m} \otimes N_{2,m} \otimes \cdots \otimes N_{k,m}$. ◻

2.1.1 Identity testing results

For noncommutative ABPs, Raz and Shpilka obtained a deterministic polynomial-time algorithm for identity testing [RS05].

Theorem 9 (Raz-Shpilka [RS05])**.** *Given as input a noncommutative ABP of width w and d many layers computing a polynomial $f \in \mathbb{F}\langle X \rangle$, there is a deterministic* $\mathrm{poly}(w, d, n)$ *time algorithm to test whether or not $f \equiv 0$.*

In fact, the following corollary is standard by now. This was first formally observed in [AMS10] using a minor adaptation of [RS05].

Corollary 10. *Given a noncommutative ABP of width w and d many layers computing a nonzero polynomial $f \in \mathbb{F}\langle X \rangle$, there is a deterministic* $\mathrm{poly}(w, d, n)$ *time algorithm which outputs a nonzero monomial m in f. If $\mathbb{F} = \mathbb{Q}$, the bit complexity of the algorithm is* $\mathrm{poly}(w, d, n, b)$ *where b is the maximum bit complexity of any coefficient in the input ABP.*

Essentially, the algorithm of Raz and Shpilka maintains basis vectors (indexed by at most w monomials) in each layer of the ABP using simple linear algebraic computations. The entries of the basis vectors are the coefficients of the indexing monomials in different nodes of that layer of the ABP.

Given such a monomial $m = x_{i_1} x_{i_2} \ldots x_{i_d}$, [AMS10] introduced a simple trick to produce a matrix tuple in $\mathrm{Mat}_{d+1}(\mathbb{F})^n$ on which f evaluates to nonzero. To see that consider a $d+1$ state deterministic finite automaton $\mathcal{A}$ that accepts only the string $x_{i_1} x_{i_2} \ldots x_{i_d}$ over the alphabet $\{x_1, x_2, \ldots, x_n\}$. The transition matrix tuple $(M_{x_1}, \ldots, M_{x_n})$ of $\mathcal{A}$ have the property that $f(M_{x_1}, \ldots, M_{x_n}) \neq 0$. More precisely, the automaton $\mathcal{A}$ is the following.

$$q_0 \xrightarrow{x_{i_1}} q_1 \xrightarrow{x_{i_2}} q_2 \xrightarrow{\cdots\cdots \quad x_{i_d}} q_d$$

The transition matrices $M_{x_j} : 1 \leqslant j \leqslant n$ are $(d+1)$ dimensional $(0,1)$-matrices with the property that $M_{x_j}(\ell, \ell+1) = 1$ if and only if x_j is the edge label between q_ℓ and $q_{\ell+1}$ for $0 \leqslant \ell \leqslant d-1$. This we record as a corollary.

Corollary 11. *Given a noncommutative ABP of width w and d layers computing a nonzero polynomial $f \in \mathbb{F}\langle X \rangle$, there is a deterministic polynomial-time algorithm that can output a matrix tuple $(M_1, M_2, \ldots, M_n)$ of dimension at most $d+1$ such that $f(M_1, M_2, \ldots, M_n) \neq 0$.*

2.1.2 Homogenization

A noncommutative (or commutative ABP) over variable set X can be easily homogenized using standard ideas. The standard reference is the survey by Shpilka and Yehudayoff [SY10, Chapter 2]. This is also explained in [RS05, Lemma 2]. We observe that the same homogenization extends to partially commutative ABPs defined over the variable set $X_{[k]}$ in the following sense. This result is field independent.

Lemma 12. *Let $f \in \mathbb{F}\langle X_{[k]}\rangle$ be a partially commutative polynomial of degree d computed by an ABP of size s. Then for any $1 \leqslant j \leqslant k$, we can efficiently homogenize the ABP over the variable set X_j, and the coefficients are also computed by ABPs over $\mathbb{F}\langle X_{[k]\setminus\{j\}}\rangle$ of size $O(sd)$.*

Proof. W.l.o.g, we describe the homogenization w.r.t. the variable set X_1. The construction is standard and we provide a sketch. Every node v is replaced by a set of nodes $(v, 0), (v, 1), \ldots, (v, d)$ where the node (v, j) computes the j^{th} homogenized component of the polynomial computed at the node v. Let $v \to u$ be an edge in the ABP labeled by $L' + L$ where L is a linear form over X_1 and L' is an affine linear form over $X_2 \sqcup \cdots \sqcup X_k$. Then we connect (v, i) to (u, i) with a label L' for $0 \leqslant i \leqslant d$. Similarly, we connect (v, i) to $(u, i + 1)$ with a label L.

The next step is to get rid of edges that labeled with affine linear forms over $\mathbb{F}\langle X_2, \ldots, X_k\rangle$. This process is repeated layer by layer starting from the source vertex at the left most layer. Suppose that there is an edge between $(v, i) \to (u, i)$ labeled with L' over $\mathbb{F}\langle X_2, \ldots, X_k\rangle$. For an edge $(w, i - 1) \to (v, i)$ already labeled with an ABP g will be changed to $(w, i - 1) \to (u, i)$ with the label $g \cdot L'$. If there is already an edge between $(w, i - 1)$ to (u, i) with a label g' which is an ABP, we update the edge label $(w, i - 1) \to (u, i)$ by $g \cdot L' + g'$. We repeat this process until we get rid of all the edges carrying affine linear forms over $\mathbb{F}\langle X_{[k]\setminus\{1\}}\rangle$. Clearly each of the ABPs on the edges are of size $O(sd)$. ◻

2.2 Cyclic Division Algebras

A division algebra D is an associative algebra over a (commutative) field $\mathbb{F}$ such that all nonzero elements in D are units (they have a multiplicative inverse). In the context of this paper, we are interested in finite-dimensional division algebras. Specifically, we focus on cyclic division algebras and their construction [Lam01, Chapter 5].

We describe the construction over $\mathbb{F} = \mathbb{Q}$. Let $F = \mathbb{Q}(z)$, where z is a commuting indeterminate. Let ω be an ℓ^{th} primitive root of unity. To be specific, let $\omega = e^{2\pi i/\ell}$. Let $K = F(\omega) = \mathbb{Q}(\omega, z)$ be the cyclic Galois extension of F obtained by adjoining ω. The elements of K are polynomials in ω (of degree at most $\ell - 1$) with coefficients from F.

Define $\sigma : K \to K$ by letting $\sigma(\omega) = \omega^k$ for some k relatively prime to ℓ and stipulating that $\sigma(a) = a$ for all $a \in F$. Then σ is an automorphism of K with F as fixed field and it generates the Galois group $\mathrm{Gal}(K/F)$.

The division algebra $D = (K/F, \sigma, z)$ is defined using a new indeterminate x as the ℓ-dimensional vector space:

$$D = K \oplus Kx \oplus \cdots \oplus Kx^{\ell-1},$$

where the (noncommutative) multiplication for D is defined by $x^\ell = z$ and $xb = \sigma(b)x$ for all $b \in K$. Then D is a division algebra of dimension ℓ^2 over F [Lam01, Theorem 14.9].

Definition 13. The *index* of D is defined to be the square root of the dimension of D over F. In our example, D is of index ℓ.

The elements of D has matrix representation in $K^{\ell\times\ell}$ from its action on the basis $\mathcal{X} = \{1, x, \ldots, x^{\ell-1}\}$. I.e., for $a \in D$ and $x^j \in \mathcal{X}$, the j^{th} row of the matrix representation is obtained by writing $x^j a$ in the $\mathcal{X}$-basis.

For example, the matrix representation $M(x)$ of x is:

$$M(x)[i,j] = \begin{cases} 1 & \text{if } j = i+1, i \leqslant \ell - 1 \\ z & \text{if } i = \ell, j = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$M(x) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ z & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

For each $b \in K$ its matrix representation $M(b)$ is:

$$M(b)[i,j] = \begin{cases} b & \text{if } i = j = 1 \\ \sigma^{i-1}(b) & \text{if } i = j, i \geqslant 2 \\ 0 & \text{otherwise.} \end{cases}$$

$$M(b) = \begin{bmatrix} b & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma(b) & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2(b) & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^{\ell-2}(b) & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^{\ell-1}(b) \end{bmatrix}$$

Remark 14. We note that $M(x)$ has a "circulant" matrix structure and $M(b)$ is a diagonal matrix. For a vector $v \in K^\ell$, it is convenient to write $\mathrm{circ}(v_1, v_2, \ldots, v_\ell)$ for the $\ell \times \ell$ matrix with $(i, i+1)^{th}$ entry v_i for $i \leqslant \ell - 1$, $(\ell, 1)^{th}$ entry as v_ℓ and remaining entries zero. Thus, we have $M(x) = \mathrm{circ}(1, 1, \ldots, 1, z)$. Similarly, we write $\mathrm{diag}(v_1, v_2, \ldots, v_\ell)$ for the diagonal matrix with entries v_i.

Fact 15. *The F-algebra generated by $M(x)$ and $M(b), b \in K$ is an isomorphic copy of the cyclic division algebra in the matrix algebra $\mathrm{Mat}_\ell(K)$.*

Proposition 16. *For all $b \in K$, $\mathrm{circ}(b, \sigma(b), \ldots, z\sigma^{\ell-1}(b)) = M(b) \cdot M(x)$.*

Define $C_{i,j} = M(\omega^{j-1}) \cdot M(x^{i-1})$ for $1 \leqslant i, j \leqslant \ell$. Observe that, $\mathfrak{B} = \{C_{ij}, i, j \in [\ell]\}$ be a F-generating set for the division algebra D. The following proposition is a standard fact.

Proposition 17. *[Lam01, Section 14(14.13)] Then K linear span of $\mathfrak{B}$ is the entire matrix algebra $\mathrm{Mat}_\ell(K)$.*

2.3 Partially Commutative Rational Series

In the following lemma, we prove that the zero testing of a series defined over partially commutative variables can be reduced to the zero testing of a polynomial of low degree. This extends such a result known for $k = 1$ [Eil74, Corollary 8.3, Page 145] (Also, see [DK21, Example 8.2, Page 23]) to the partially commutative setting where $X = X_{[k]}$. The proof is linear algebraic and we crucially use the fact that the partially commutative ring $\mathbb{F}\langle X_{[k]} \rangle$ is embedded in the universal skew field $\mathfrak{U}_{[k]}$ [KVV20] as mentioned in Section 1 (a formal statement regarding the construction of $\mathfrak{U}_{[k]}$ is given in Theorem 21). In [Wor13, Proposition 5], Worrell has proved the same result using Ore domains.

Lemma 18. *Consider the universal skew field $\mathfrak{U}_{[k]}$ over $\mathbb{F}\langle X_{[k]}\rangle$. Let $L \in \mathfrak{U}_{[k]}^{s\times s}$ be a linear matrix over $X_{[k]}$, u, v are $1\times s$ and $s\times 1$ dimensional vectors whose entries are linear forms over $X_{[k]}$. Then $u\left(\sum_{i\geqslant 0} L^i\right)v = 0$ if and only if $u\left(\sum_{i\leqslant s} L^i\right)v = 0$.*

Proof. If $u\left(\sum_{i\geqslant 0} L^i\right)v = 0$ then clearly $u\left(\sum_{i\leqslant s} L^i\right)v = 0$ since different homogeneous components will not mix together.

To see the other direction, we first note that $u\left(\sum_{i\leqslant s} L^i\right)v = 0$ implies $uL^iv = 0, 0 \leqslant i \leqslant s$ as each term in the sum is a different homogeneous part. Now consider the $s+1$ many vectors $v_i = u\cdot L^i$ for $0 \leqslant i \leqslant s$. Since each v_i is in the left $\mathfrak{U}_{[k]}$-module U^s, and $\mathfrak{U}_{[k]}$ is a (skew) field, they cannot all be $\mathfrak{U}_{[k]}$-linearly independent. That means there are $\lambda_0, \ldots, \lambda_s$ in $\mathfrak{U}_{[k]}$, not all zero, such that the *left* linear combination $\sum_{j=0}^{s}\lambda_j v_j = \sum_{j=0}^{s}\lambda_j uL^j = 0$. Let t be the largest index such that λ_t is nonzero. Then we can write $u\cdot L^t = -\sum_{j=0}^{t-1}\lambda_t{}^{-1}\lambda_j u\cdot L^j$. Multiplying both sides on the right by L^{s+1-t}, we obtain

$$u\cdot L^{s+1} = -\sum_{j=0}^{t-1}\lambda_t{}^{-1}\lambda_j u\cdot L^{j+s+1-t}.$$

But this will imply that $u\cdot L^{s+1}\cdot v = 0$ since $u\cdot L^{j+s+1-t}v = 0$ for $j \leqslant t-1$. Now, assuming inductively that $uL^iv = 0$ for some $i \geqslant s+1$, we can similarly prove that $uL^{i+1}v = 0$. It follows that the entire series is zero. □

2.4 Equivalent Notions of Matrix Rank

We first recall the definition of the noncommutative rank of a linear matrix in noncommutative variables, the computationally useful notion of its blow-up rank, and their equivalence.[5]

Definition 19. The noncommutative rank (ncrank) of an $s\times s$ linear matrix T over the noncommuting variables $x_1, x_2, \ldots, x_n$ is equal to the size of the largest invertible (square) submatrix of T.

Let T be an $s\times s$ matrix whose entries are affine linear forms over $\{x_1, x_2, \ldots, x_n\}$. We can write $T = A_0 + \sum_{i=1}^{n} A_i x_i$ where $A_0, A_1, \ldots, A_n$ are the coefficient matrices. Given matrix T, for $d \in \mathbb{N}$ we define the set of "blow-up" matrices

$$T^{\{d\}} \;=\; \{T(\underline{M}) \mid \underline{M} \in \mathrm{Mat}_d(\mathbb{F})^n\},$$

where $T(\underline{M}) = A_0 \otimes I_d + \sum_{i=1}^{n} A_i \otimes M_i$. Then we define the *blow-up rank* of T at d as $\mathrm{rank}(T^{\{d\}}) = \max_{\underline{M}}\{\mathrm{rank}(T(\underline{M}))\}$. The regularity lemma [IQS17, DM17, IQS18] shows that $\mathrm{rank}(T^{\{d\}})$ is always a multiple bd of d. Thus, we can define the *blow-up rank* of T as b, which the largest positive integer such that for some d we have $\mathrm{rank}(T^{\{d\}}) = bd$. The regularity lemma also implies that the blow-up rank of T is precisely $\mathrm{ncrank}(T)$.

Fact 20. *For a linear matrix T in noncommutative variables* $\mathrm{ncrank}(T)$ *is its blow-up rank.*

The blow-up rank is algorithmically useful [IQS17, DM17, IQS18]. In Section 3.1, we will discuss this aspect further. Let us now consider a set $X = X_{[k]}$ of partially commutative variables and T be a linear matrix with linear forms over $\mathbb{F}\langle X_{[k]}\rangle$. The main result of [KVV20] is stated in the following theorem.

[5] We note that Cohn's text [Coh95] has a detailed discussion of matrix rank over general noncommutative rings.

Theorem 21. *[KVV20, Theorem 1.1] For arbitrary $k \in \mathbb{N}$, the ring $\mathbb{F}\langle X_{[k]}\rangle$ can be embedded in a universal skew field of fractions $\mathfrak{U}_{[k]}$.*

As a consequence of the above theorem and some properties of noncommutative rings [Coh95], we can define the rank of matrices over $\mathbb{F}\langle X_{[k]}\rangle$ for a partially commutative variable set $X_{[k]}$ as follows.

Definition 22. The partially commutative rank (pc-rank) of an $s \times s$ linear matrix T over the partially commutative variable set $X_{[k]}$ is equal to the size of the largest invertible (over $\mathfrak{U}_{[k]}$) square submatrix of T.

The following crucial result is also shown in [KVV20].

Proposition 23. *[KVV20, Proposition 3.8] A matrix T is invertible over $\mathfrak{U}_{[k]}$ if and only if there exists matrix substitutions for the variables $x \in X_i : 1 \leqslant i \leqslant k$ of the form*

$$I_{d_1} \otimes I_{d_2} \otimes \cdots \otimes I_{d_{i-1}} \otimes M_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k} \tag{5}$$

such that T evaluates to an invertible matrix. Here M_x is a $d_i \times d_i$ matrix and $d_1, \ldots, d_k \in \mathbb{N}$.

The above proposition in fact gives us the right analogue of blow-up rank for the partially commutative ring $\mathbb{F}\langle X_{[k]}\rangle$, and is crucial for our algorithms. Since we will be using partially commutative matrix substitutions of the above kind for variables in $X_{[k]}$ for rank computations in this paper, we introduce the following useful definition.

Definition 24. We call the matrix substitution of the form given in the expression 5 as a type-i k-fold tensor product. Also $\underline{d} = (d_1, d_2, \ldots, d_k)$ is the shape of the tensor.

Thus, for a linear matrix T over $X_{[k]}$ we seek type-i matrix substitutions for variables in X_i for each i, which are $(d_1, d_2, \ldots, d_k)$ shape tensor products for a suitable choice of the dimensions $d_i, 1 \leqslant i \leqslant k$.

3 An Algorithm for NSINGULAR based on NC-PIT

The key ideas for the proofs of Theorems 1 and 3 come from the design of a somewhat simpler algorithm for NSingular (which is the case for $k = 1$) that we discuss in this section. As explained earlier, the algorithm in [IQS18] has two main steps: rank increment, rounding and blow-up control. In the simpler algorithm, rounding and blow-up control is essentially the same as in [IQS18]. But the rank increment step is quite different. It is based on an efficient reduction to the noncommutative ABP identity testing. This connection extends to the partially commutative setting and plays a crucial role in the proofs of Theorems 1 and 3. Motivated by Fact 20, we give the following definition. We fix the field to be $\mathbb{Q}$.

Definition 25 (Witness of ncrank r)**.** Let $A_0, A_1, \ldots, A_n \in \mathrm{Mat}_s(\mathbb{Q})$ and $T = A_0 + \sum_{i=1}^{n} A_i x_i$. We say that $\underline{p} = (p_1, \ldots, p_n) \in \mathrm{Mat}_d(\mathbb{Q})^n$ for some d is a witness of noncommutative rank (at least) r of T, if $\mathrm{rank}(T(\underline{p})) \geqslant rd$.

3.1 Constructive Regularity Lemma

Suppose that for a linear matrix T, we already have a matrix tuple $\underline{q}$ over $\mathrm{Mat}_d(\mathbb{Q})$, a witness of rank r of T such that $\mathrm{rank}(T(\underline{q})) > rd$. Then the constructive regularity lemma offers a simple and general procedure to get a $d \times d$ witness of rank $r+1$ for T [IQS18]. We present essentially the same proof as described in [IQS18]. But for clarity and for setting the context of the main results in the next section, we use the explicit cyclic division algebra construction described in Section 2.2. Following Section 2.2, the field $F = \mathbb{Q}(z)$ and $K = F(\omega)$.

Lemma 26. *[IQS18] For any $s \times s$ matrix $T = A_0 + \sum_{i=1}^{n} A_i x_i$, and a matrix tuple $\underline{q} = (q_1, \ldots, q_n) \in \mathrm{Mat}_d(\mathbb{Q})^n$ such that $\mathrm{rank}(T(\underline{q})) > rd$, there exists a deterministic $\mathrm{poly}(n, s, d)$-time algorithm that returns another matrix substitution $\underline{q}' = (q'_1, \ldots, q'_n) \in \mathrm{Mat}_d(\mathbb{Q})^n$ such that $\mathrm{rank}(T(\underline{q}')) \geqslant (r+1)d$.*

Proof. Let $D = (K/F, \sigma, z)$ be the cyclic division algebra described in Section 2.2. Recall that $\mathfrak{B} = \{C_{i,j} : i, j \in [d]\}$ is a F-generating set of D.

1. By Proposition 17, we can express $q_k = \sum_{i,j} \lambda_{i,j,k} C_{i,j}$ where $\lambda_{i,j,k}$, $1 \leqslant k \leqslant n$ are unknown variables which take values in K. A linear algebraic computation yields the values $\lambda^0_{i,j,k}$ where $1 \leqslant i, j \leqslant \ell$, and $1 \leqslant k \leqslant n$ for the unknowns in K.

2. Now the goal is to compute a $d \times d$ tuple $\underline{q}'' = (q''_1, \ldots, q''_n)$ such that $q''_k = \sum_{i,j} \mu^0_{i,j,k} C_{i,j}$ where $\mu^0_{i,j,k} \in \mathbb{Q}$ and $\mathrm{rank}(T(\underline{q}'')) \geqslant (r+1)d$. We briefly describe the procedure outlined in [IQS18]. Write $\tilde{q}_1 = \mu_{1,1,1} C_{1,1,1} + \sum_{(i,j) \neq (1,1)} \lambda^0_{i,j,1} C_{i,j}$ where $\mu_{1,1,1}$ is a variable. There will be a sub-matrix of size $> rd$ whose minor is non-zero, under the current substitution $(\tilde{q}_1, q_2, \ldots, q_n)$. Since the determinant of that sub-matrix is a univariate polynomial in $\mu_{1,1,1}$ and degree $\mathrm{poly}(r, d)$, we can easily fix the value of $\mu_{1,1,1}$ from $\mathbb{Q}$ such that the minor remains nonzero. Repeating the procedure, we can compute a tuple $\underline{q}''$. Since $\underline{q}''$ is a tuple over the division algebra, $\mathrm{rank}(T(\underline{q}'')) \geqslant (r+1)d$.

The last line of the above proof is easy to see. The matrix $T(\underline{q}'')$ can be viewed as a $s \times s$ block-matrix of d-dimensional blocks, and each such block is an element in D. Since Gaussian elimination is supported over division algebras, up to elementary row and column operations, we can transform $T(\underline{q}'')$ as:

$$\left(\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right)$$

where I is an identity matrix which has at least $r+1$ blocks of identity matrices I_d on its diagonal. Hence $\mathrm{rank}(T(\underline{q}'')) \geqslant (r+1)d$. From the tuple $\underline{q}''$, we can easily obtain the desired tuple $\underline{q}'$ as follows. We can think ω and z as fresh commutative parameters t_1, t_2. Clearly, after the substitution the determinant of that $(r+1)d$ dimensional submatrix is a bivariate polynomial in t_1, t_2 of degree $\leqslant (r+1)d$. We can set the variables from a set of size $O(rd)$ in $\mathbb{Q}$ such that the submatrix remains invertible. Replacing the variables t_1, t_2 by such values over $\mathbb{Q}$, we get the tuple $\underline{q}'$ defined over $\mathbb{Q}$. ◻

3.2 Rank Increment Step

This is quite different from the rank increment step in [IQS18]. More importantly, this turns out to be readily extendable to the partially commutative case in the proof of Theorem 1. The increment step gradually constructs a witness at every stage. Given a *witness of rank r* for T, the algorithm checks if r is the maximum possible rank. If not, it produces a witness of rank at least $r+1$ by solving an instance of ABP identity testing and iterates. At a high level, it has a conceptual similarity with the idea used in [BBJP19] in approximating commutative rank.

For an $s \times s$ linear matrix $T(\underline{x}) = A_0 + \sum_{i=1}^{n} A_i x_i$ and $d \in \mathbb{N}$, define

$$T_d(Z) = A_0 \otimes I_d + \sum_{i=1}^{n} A_i \otimes Z_i$$

where $Z_i = (z_{jk}^{(i)})_{1 \leqslant j,k \leqslant d}$ is a $d \times d$ generic matrix with noncommutative indeterminates. In other words, $Z = (Z_1, Z_2, \ldots, Z_n)$ is the substitution used for the variables $x_1, x_2, \ldots, x_n$ in T. Now $T_d(Z)$ is a linear matrix of dimension sd over the variables $\{z_{jk}^{(i)}\}_{1 \leqslant j,k \leqslant d, 1 \leqslant i \leqslant n}$.

Remark 27. It is immediate to see that any $d \times d$ matrix shift $T_d(Z_1 + p_1, Z_2 + p_2, \ldots, Z_n + p_n)$ is indeed a scalar shift for the variables $\{z_{jk}^{(i)}\}_{1 \leqslant j,k \leqslant d, 1 \leqslant i \leqslant n}$ in the matrix T_d.

Lemma 28. $\mathrm{ncrank}(T_d) = d \cdot \mathrm{ncrank}(T)$.

Proof. Let $\mathrm{ncrank}(T) = r$. Then, for every sufficiently large d'', the maximum rank obtained by evaluating T over all $d'' \times d''$ matrix tuples is rd''. Let $d'' = dd'$ be a multiple of d and let $\underline{q} = (q_1, \ldots, q_n)$ be a matrix tuple such that $\mathrm{rank}(T(\underline{q})) = rdd'$. Let

$$\underline{p} = (p_{11}^{(1)}, \ldots, p_{dd}^{(1)}, \ldots, p_{11}^{(n)}, \ldots, p_{dd}^{(n)})$$

be the matrix tuple such that each $q_i = (p_{jk}^{(i)})_{1 \leqslant j,k \leqslant d}$. That is, we think of q_i as the $d \times d$ block matrix where the $(j,k)^{th}$ block is $p_{jk}^{(i)}$. Notice that $T_d(\underline{p}) = A_0 \otimes I_{dd'} + \sum_{i=1}^{n} A_i \otimes q_i = T(\underline{q})$, with the matrix q_i substituted for the variable x_i in T. Therefore, $\mathrm{rank}(T(\underline{q})) = \mathrm{rank}(T_d(\underline{p}))$ and $\mathrm{ncrank}(T_d) \geqslant rd$.

For the other direction, as $\mathrm{ncrank}(T) = r$, we can write $T = PQ$ where P, Q are $s \times r$ and $r \times s$ matrices respectively with linear entries [Coh95]. We can now define an $sd \times rd$ matrix $P'(Z)$ by substituting each x_i by Z_i in the matrix $P(\underline{x})$. Similarly, we can define a $rd \times sd$ matrix $Q'(Z)$ from $Q(\underline{x})$. Notice that, $T_d = P'Q'$. Therefore, $\mathrm{ncrank}(T_d) \leqslant rd$. Hence, the lemma follows. ◻

3.2.1 A noncommutative ABP identity testing reduction step

Suppose, now, that we have computed a witness of noncommutative rank r of T, namely $\underline{p} = (p_1, \ldots, p_n) \in \mathrm{Mat}_d(\mathbb{Q})^n$ (by construction, we will ensure that $d \leqslant r+1$). We will now describe how to check whether $\mathrm{ncrank}(T) > r$ or not. Observe that

$$T_d(Z_1 + p_1, \ldots, Z_n + p_n) = U \left(\begin{array}{c|c} I_{rd} - L & A \\ \hline B & C \end{array} \right) V$$

for invertible transformations U, V in $\mathrm{Mat}_{rd}(\mathbb{Q})$. In fact, applying further invertible transformations U', V', we can write

$$T_d(Z_1 + p_1, \ldots, Z_n + p_n) = UU' \left(\begin{array}{c|c} I_{rd} - L & 0 \\ \hline 0 & C - B(I_{rd} - L)^{-1} A \end{array} \right) V'V.$$

$$\text{Here, } U' = \left(\begin{array}{c|c} I_{rd} & 0 \\ \hline B(I_{rd}-L)^{-1} & I_{(s-r)d} \end{array}\right), \qquad V' = \left(\begin{array}{c|c} I_{rd} & (I_{rd}-L)^{-1}A \\ \hline 0 & I_{(s-r)d} \end{array}\right).$$

Let $\widetilde{T_d} = C - B(I_{rd} - L)^{-1}A$. The entries in C, B, L, A are linear forms over the variables X. Notice that the $(i, j)^{th}$ entry of $\widetilde{T_d}$ is given by $\widetilde{(T_d)}_{ij} = C_{ij} - B_i(I_{rd} - L)^{-1}A_j$ where B_i is the i^{th} row vector of B and A_j is the j^{th} column vector of A.

Lemma 29. ncrank$(T) > r$ *if and only if* $\widetilde{(T_d)}_{ij} \neq 0$ *for some choice of* i, j.

Proof. Let ncrank$(T) > r$. Then by Lemma 28, ncrank$(T_d) > rd$. The noncommutative rank of a linear matrix is invariant under a scalar shift [6], hence ncrank$(T_d(Z_1+p_1, \ldots, Z_n+p_n)) =$ ncrank$(T_d) > rd$. However, if $C - B(I_{rd} - L)^{-1}A$ is a zero matrix, this is impossible.

Conversely if $\widetilde{(T_d)}_{ij} = C_{ij} - B_i(I_{rd} - L)^{-1}A_j$ is nonzero for some indices i, j, we can find matrix substitutions $\tilde{p}^{(k)}_{\ell_1\ell_2}$ of dimension d' for the variables $\{z^{(k)}_{\ell_1\ell_2}\}_{1\leqslant \ell_1,\ell_2\leqslant d, 1\leqslant k\leqslant n}$, such that the rank of $T_d(Z_1+p_1, \ldots, Z_n+p_n)$ on that substitution is more than rdd'. Therefore, ncrank$(T_d(Z_1+p_1, \ldots, Z_n+p_n)) > rd$. Hence ncrank$(T_d) > rd$. By Lemma 28, we get that ncrank$(T) > r$. ◻

Now, applying Lemma 18 for $k = 1$ we note that the infinite series $\widetilde{(T_d)}_{ij} \neq 0$ if and only if the truncated polynomial

$$\widetilde{P}_{ij} = C_{ij} - B_i \left(\sum_{k\leqslant rd} L^k\right) A_j \neq 0. \tag{6}$$

To see that Lemma 18 is applicable above, notice that the C_{ij} is a linear form and $B_i\left(\sum_{k\leqslant rd} L^k\right)A_j$ generates terms of degree at least 2.

Next, we apply Corollary 10 and Corollary 11 to output a matrix tuple efficiently on which $\widetilde{(T_d)}_{ij}$ evaluates to nonzero and $I_{rd} - L$ evaluates to a full rank matrix.

Lemma 30. *There is a deterministic* poly(n, r, d)*-time algorithm that can output a matrix tuple* $\underline{q}$ *of dimension at most* $d' = 2rd$ *for the* Z *variables such that* $I_{rdd'} - L(\underline{q})$ *is invertible and* $\widetilde{(T_d)}_{ij}(\underline{q}) \neq 0$.

Proof. Notice that $\widetilde{P}_{ij}$ is an ABP of size poly(r, d) and the number of layers is at most $rd + 1$. Applying Corollary 11, we get a matrix tuple of dimension at most $rd + 2$ such that $\widetilde{P}_{ij}$ evaluates on it to nonzero. By simple padding, we can get a matrix tuple $\underline{q}'$ of dimension $d' = 2rd$ such that $\widetilde{P}_{ij}(\underline{q}') \neq 0$. Since $\underline{q}'$ is a substitution for the Z variables $\{z^{(k)}_{\ell_1\ell_2}\}$ where $1 \leqslant k \leqslant n, 1 \leqslant \ell_1, \ell_2 \leqslant d$, we write $\underline{q}' = (q'^{(1)}_{11}, \ldots, q'^{(1)}_{dd}, \ldots, q'^{(n)}_{11}, \ldots, q'^{(n)}_{dd})$ for more clarity. Here each $q'^{(k)}_{\ell_1\ell_2}$ is a d' dimensional matrix.

Consider a commutative variable t and the scaled matrix tuple $t\underline{q}'$. It is easy to see that the infinite series $C_{ij} - B_i(I_{rdd'} - L(t\underline{q}'))^{-1}A_j$ is nonzero since the k^{th} homogeneous part $t^k B_i L^k(\underline{q}')A_j$ will not mix with other homogeneous components.

However this also has a rational representation $\widetilde{(T_d)}_{ij}(t\underline{q}') = \gamma_1(t)/\gamma_2(t)$ where t-degrees of the polynomials $\gamma_1(t), \gamma_2(t)$ are bounded by rdd'. Moreover, $I_{rdd'} - L(t\underline{q}')$ is an invertible matrix and the

[6] Suppose a linear matrix L achieves the maximum rank at matrix substitution $\underline{q}$ of some dimension d. Then, for any scalar shift $(\alpha_1, \ldots, \alpha_n)$, the linear matrix $L(\underline{x} + \underline{\alpha})$ achieves the same rank at the matrix substitution $\underline{q} - \underline{\alpha} \otimes I_d$.

degree of $\det(I_{rdd'} - L(t\underline{q}'))$ is bounded by rdd' over the variable t. Simply by varying the variable t over a suitable large set Γ of size $O(rd)$, we can fix a value for $t = t_0$ such that $(\widetilde{T_d})_{ij}(t_0\underline{q}') \neq 0$ and $I_{rdd'} - L(t_0\underline{q}')$ is of rank rdd'. Define $\underline{q} = t_0\underline{q}'$. □

Following is an immediate corollary.

Corollary 31. *Suppose Lemma 30 outputs a matrix tuple $\underline{q}$. We can compute another matrix tuple $\underline{p}'$ of dimension dd' which is a witness of* $\operatorname{ncrank}(T) > r$.

Proof. Define the matrix tuple $\underline{q}'' = (q''^{(1)}_{11}, \dots, q''^{(1)}_{dd}, \dots, q''^{(n)}_{11}, \dots, q''^{(n)}_{dd})$ where $q''^{(k)}_{\ell_1\ell_2} = q^{(k)}_{\ell_1\ell_2} + p^{(k)}_{\ell_1\ell_2} \otimes I_{d'}$ is a d' dimensional matrix tuple for $1 \leqslant k \leqslant n, 1 \leqslant \ell_1, \ell_2 \leqslant d$.

Lemma 30 shows that the rank of T_d evaluated on the matrix tuple $\underline{q}''$ is more than rdd'. This is same as saying that $T_d(Z)$ is of rank more than rdd' when the variable $z^k_{\ell_1,\ell_2} : 1 \leqslant k \leqslant n, 1 \leqslant \ell_1, \ell_2 \leqslant d$ is substituted by $q''^{(k)}_{\ell_1\ell_2}$. Hence $\operatorname{ncrank}(T_d) > rd$. By Lemma 28, we know that $\operatorname{ncrank}(T) > r$. Moreover, we obtain a matrix tuple $\underline{p}' = (p'_1, p'_2, \dots, p'_n)$ which is a witness of $\operatorname{ncrank}(T) > r$, where $p'_k = \left(q''^{(k)}_{\ell_1\ell_2}\right)_{1 \leqslant \ell_1, \ell_2 \leqslant d} : 1 \leqslant k \leqslant n$. Notice that $\underline{p}'$ is the substitution for the $\underline{x}$ variables. □

3.2.2 Rounding and blow-up Control

Next, we apply Lemma 26 which gives a rounding procedure to get a matrix tuple of dimension $d_1 = dd'$ to witness that $\operatorname{ncrank}(T) = r'$ where $r' \geqslant r + 1$. Call that new matrix tuple as $\underline{p}''$.

However, we cannot afford to have such a dimension blow-up for the witness matrix tuple in every step of the iteration as it incurs an exponential blow-up in the dimension of the final witness. To control that, we use a simple trick from [IQS18] which we describe for the sake of completeness.

Lemma 32. *Consider an $s \times s$ linear matrix T and a matrix tuple $\underline{p}''$ in* $\mathrm{Mat}_{d_1}(\mathbb{Q})^n$ *such that $\underline{p}''$ is a witness of rank r' of T. We can efficiently compute another matrix tuple $\widehat{\underline{p}}$ of dimension at most $r' + 1$ (over $\mathbb{Q}$) such that $\widehat{\underline{p}}$ is also a witness of rank r' of T.*

Proof. Consider a sub-matrix A in $T(\underline{p}'')$ such that $\operatorname{rank}(A)$ is at least $r'd_1$. From each matrix in the tuple $\underline{p}''$, remove the last row and the column to get another tuple $\widetilde{\underline{p}}$. We claim that the corresponding sub-matrix A' in $T(\widetilde{\underline{p}})$ is of rank $> (r'-1)(d_1-1)$ as long as $d_1 > r' + 1$. Otherwise, $\operatorname{rank}(A) \leqslant \operatorname{rank}(A') + 2r' \leqslant (r'-1)(d_1-1) + 2r' = r'd_1 - d_1 + r' + 1 < r'd_1$. Now we can use the constructive regularity lemma (Lemma 26) on the tuple $\widetilde{\underline{p}}$ to obtain another witness of dimension $d_1 - 1$ which is a witness of rank r' of T. Applying the procedure repeatedly, we can control the blow-up in the dimension within $r' + 1$ and get the witness tuple $\widehat{\underline{p}}$. □

3.3 The Algorithm for NSINGULAR

We formally state the main steps of the algorithm.

Algorithm for NSINGULAR

Input: $T = A_0 + \sum_{i=1}^n A_i x_i$ where $A_0, A_1, \dots, A_n \in \mathrm{Mat}_s(\mathbb{Q})$.

Output: The noncommutative rank of T and a set of matrix assignments that witness $\operatorname{ncrank}(T)$.

The algorithm gradually increases the rank and finds a witness for it. Suppose at any intermediate stage, we already have a matrix tuple $\underline{p}$ in $\mathrm{Mat}_d(\mathbb{Q})^n$, a *witness of rank* r of T.

1. (Is r the maximum rank?) Use Theorem 9 to check whether the polynomial $\widetilde{P}_{ij} \neq 0$ (as defined in Equation 6) for some choice of i, j.
2. If no such choice for i, j can be found, then STOP and output r to be the noncommutative rank of T.
3. (Otherwise, construct a witness of rank $r+1$ and repeat Step 1) We implement the following steps to construct a rank $(r+1)$-witness:
   (a) [Rank increment step] Apply Corollary 31 to find a $d_1 \times d_1$ matrix substitution $\underline{p}' = (p'_1, \ldots, p'_n)$ such that $\mathrm{rank}(T(\underline{p}')) > rd_1$ where $d_1 = 2rd^2$.
   (b) [Rounding using the regularity lemma] Apply Lemma 26 to find another $d_1 \times d_1$ matrix substitution $(p''_1, \ldots, p''_n)$ such that the rank of T evaluated at $(p''_1, \ldots, p''_n)$ is $r'd_1$ where $r' \geqslant r+1$.
   (c) [Reducing the witness size] Apply Lemma 32 to find a matrix substitution $\underline{\widehat{p}} = (\widehat{p}_1, \ldots, \widehat{p}_n)$ of dimension $d' \leqslant r'+1$, such that the rank of T evaluated at $\underline{\widehat{p}}$ is $\geqslant r'd'$.

Next we analyze the performance of the algorithm.

Analysis Since the noncommutative rank of T is at most s, the algorithm iterates at most s steps. Lemma 18 (for $k=1$), Theorem 9, and Lemma 30 guarantee that Step 1 and Step 3(a) can be done in $\mathrm{poly}(n, r, d)$ steps. Step 3(b) and 3(c) require straightforward linear algebraic computations discussed in Section 3.2.2 which can be performed in $\mathrm{poly}(n, d, r)$ time. Since $d \leqslant s+1$ throughout the process, the run time is bounded by $\mathrm{poly}(n, s)$.

We now explain the simple analysis of the bit complexity of the algorithm (since $\mathbb{F} = \mathbb{Q}$). Suppose the witness of rank r computed by the algorithm has bit complexity b. Notice that in the rank increment step the matrix constructed in Corollary 11 has only $0, 1$ entries and the parameter t_0 is of size $\mathrm{poly}(s, d)$. Hence, the bit complexity after step 3(a) can change to $O(b + \log(sd))$ at most. Step 3(b) is a simple linear algebraic step that can incur an additive term of $\mathrm{poly}(s, d)$ to the bit complexity. Thus, the bit complexity of the witness of rank $r+1$ is bounded by $b + \mathrm{poly}(s, d)$. Since the bit complexity for the first step is bounded by the input coefficients, it follows that the overall bit complexity of the algorithm is polynomial in s and the input size.

Remark 33. The algorithm of NSingular can be adapted over fields of positive characteristic by extending the division algebra construction over such fields [Pie82, Section 15.4]. However, since our main motivation is to prove Theorem 1 and Theorem 3, we prefer to state the algorithm for NSingular over $\mathbb{F} = \mathbb{Q}$.

4 Proofs of the Main Theorems

The goal of this section is to present the proofs of Theorems 1 and 3 by designing the subroutines PC-PIT$_k$ and PC-Rank$_k$.

The subroutine PC-PIT$_k$ takes as input an ABP $\mathcal{A}$ of size s computing a polynomial $f \in \mathbb{F}\langle X_{[k]}\rangle$. It finds substitution matrices of the form 1 for the variables $x \in X_{[k]}$ such that f evaluates to a nonzero matrix if f is a nonzero polynomial. Moreover, the dimension of the substitution matrices is a polynomial function of the input size.

The subroutine PC-RANK$_k$ takes as input a linear matrix T of size s over the set of variables $X_{[k]}$ and finds matrix assignments of the form 1 and dimension d, to the variables such that the rank of the final matrix is $d \cdot \text{pc-rank}(T)$. Moreover, the dimension d is a polynomial of the input size.

These two recursive subroutines intertwine, giving the proofs of Theorem 1 and Theorem 3. Recall that for a set $S \subseteq [k]$, X_S refers to $\bigsqcup_{i\in S} X_i$. When we use PC-RANK (resp. PC-PIT) subroutine on X_S with $|S| = \ell$, we refer it as PC-RANK$_\ell$ (resp. PC-PIT$_\ell$). Also, recall from Definition 24 that the substitution matrices of the form 1 are type-i k-fold tensors of shape $\underline{d} = (d_1, d_2, \ldots, d_k)$.

4.1 Identity testing of partially commutative ABPs

In this section, we describe the subroutine PC-PIT$_k$. Basically, given a partially commutative ABP as input with edges labeled by linear forms over $\mathbb{Q}\langle X\rangle$ where $X = X_{[k]}$, we develop a deterministic algorithm for identity testing of such ABPs.

We need to generalize the following result shown in [ACG+23] for the noncommutative case. Suppose M is a matrix over $\mathbb{F}\langle X\rangle$, for noncommutative X, such that each M_{ij} is given as input by a linear pencil (See, the definition 5). Then we can efficiently reduces rank computation of M to the rank computation of a (noncommutative) linear matrix over X.

Lemma 34. [ACG+23, Lemma 23] *Let $X = \{x_1, \ldots, x_n\}$ be a set of noncommutative variables. Let $M \in \mathbb{F}\langle X\rangle^{m\times m}$ be a matrix where each $(i,j)^{th}$ entry $M_{ij} \in \mathbb{F}\langle X\rangle$ is given as input by a size s linear pencil L_{ij}. Then, there is a polynomial-time algorithm that computes a linear matrix L of size $m^2 s + m$ such that,*

$$\text{ncrank}(L) = m^2 s + \text{ncrank}(M).$$

It is easy to see by inspection that the proof of the above lemma [ACG+23] holds even when $X_{[k]}$ is a set of partially commutative variables. More precisely, we have the following generalization, proved in the appendix.

Lemma 35. *Let $X = X_{[k]}$ be a set of partially commutative variables. Let $M \in \mathbb{Q}\langle X_{[k]}\rangle^{m\times m}$ be a matrix where each M_{ij} is given by a linear pencil L_{ij} of size s.*[7] *Then, there is a polynomial-time algorithm that computes a linear matrix L of size $m^2 s + m$ such that,*

$$\text{pc-rank}(L) = m^2 s + \text{pc-rank}(M).$$

The actual application of this lemma in the next theorem is as follows: Suppose M is an input matrix whose entries are ABPs defined over the set of partially commutative variables $X_{[k]}$. By Proposition 7, size s ABPs have linear pencils of size $O(s)$ and, moreover, the linear pencils can be computed in time poly(s). As a result, the rank computation problem for such a matrix M can be reduced in poly(s) time to rank computation of a linear matrix over $X_{[k]}$.

Theorem 36. *Given an input ABP $\mathcal{A}$ of size s, width w, computing a polynomial $f \in \mathbb{Q}\langle X_{[k]}\rangle$ of degree d, the subroutine* PC-PIT$_k(\mathcal{A}, s, w, d, X_{[k]})$ *reduces the identity testing problem for f to at most $O(ds^3)$ instances of PC-RANK$_{k-1}$ problem for linear matrices of size $O(s^5)$ and at most d recursive calls of* PC-PIT$_{k-1}$ *for an ABP of size $O(sd^2)$, width $O(sd)$, computing a polynomial of degree $\leqslant d$ in $\mathbb{Q}\langle X_{[k]\setminus\{1\}}\rangle$ in deterministic* poly(s) *time. Moreover, it finds assignments to the variables in $X_j : 1 \leqslant j \leqslant k$ which are of the form $I_{d_1} \otimes \cdots \otimes I_{d_{j-1}} \otimes M_x \otimes I_{d_{j+1}} \otimes \cdots \otimes I_{d_k}$ such that f evaluates to a nonzero matrix if f is originally a nonzero polynomial. The dimensions $d_1, d_2, \ldots, d_k$ are at most $d + 1$.*

[7] See Definition 6.

Proof. Firstly, we explain how PC-PIT$_k$ finds the substitution matrices for the variables in X_1. We view the edge labels as affine linear forms over the variables in X_1 and the coefficients are over the ring $\mathbb{Q}\langle X_{[k]\setminus\{1\}}\rangle$ inside $\mathfrak{U}_{[k]\setminus\{1\}}$ by Theorem 21.

As discussed in Section 2, the Raz-Shpilka algorithm [RS05], which is for a noncommutative set of variables X, is linear algebraic: We can assume the ABP is layered and the width is w at each layer. For each monomial m of degree j, there is a corresponding w-dimensional vector $v_m \in \mathbb{F}^w$ of m's coefficients at the w nodes in layer j. Now, the idea is to maintain a set of at most w many monomials $m_1, m_2, \ldots, m_w$ such that their corresponding vectors v_{m_i} are linearly independent and their $\mathbb{Q}$-linear span includes all such coefficient vectors v_m. Then, the Raz-Shpilka algorithm proceeds to layer $j+1$ with some linear algebraic computation.

We will broadly use the same approach for the partially commutative case. Applying the procedure discussed in the proof of Lemma 12, we first homogenize the ABP with respect to the variables in X_1. It suffices to solve the identity testing problem for such an X_1-homogenized ABP. It is easy to check that the edges of this homogenized ABP are labeled by linear forms $\sum_{i=1}^{n} \alpha_i x_i$ in variables $x_i \in X_1$, where the α_i are polynomials in $\mathbb{Q}\langle X_{[k]\setminus\{1\}}\rangle$. Moreover, each α_i is given by an ABP of size $O(sd) = O(s^2)$ by Lemma 12.

Inductively, at the j^{th} level, suppose the monomials computed are $m_1, m_2, \ldots, m_{w'}$ in X_1^j, where $w' \leqslant w$. Let the corresponding coefficient vectors be $v_1, v_2, \ldots, v_{w'}$ over the ring $\mathbb{Q}\langle X_{[k]\setminus\{1\}}\rangle$. Again by Lemma 12, entries of the v_i are given by ABPs over $X_{[k]\setminus\{1\}}$ of size $O(s^2 j)$. Moreover, the vectors $v_1, v_2, \ldots, v_{w'}$ are $\mathfrak{U}_{[k]\setminus\{1\}}$-spanning set for the coefficient vectors of monomials at layer j (to be precise, as a left $\mathfrak{U}_{[k]\setminus\{1\}}$-module).

Now, for the $(j+1)^{th}$ level, we need to compute at most w many $\mathfrak{U}_{[k]\setminus\{1\}}$-linearly independent vectors from the at most nw many coefficient vectors of the $\{m_i x_j : 1 \leqslant i \leqslant w', x_j \in X_1\}$. Clearly, this is the problem of computing the rank of these at most nw coefficient vectors whose entries are ABPs over the variables in $X_{[k]\setminus\{1\}}$. This is because, given a set of w-dimensional $\mathfrak{U}_{[k]\setminus\{1\}}$-linearly independent vectors $v'_1, \ldots, v'_{\ell'}$ and another vector v, the rank of this matrix with $\ell'+1$ columns is ℓ' precisely if v is in the $\mathfrak{U}_{[k]\setminus\{1\}}$-span of $v'_1, \ldots, v'_{\ell'}$. The columns of the matrix are $v'_1, \ldots, v'_{\ell'}, v$ and we can make it a square matrix by padding with zero columns. Applying Lemma 35, we can reduce it to the PC-Rank$_{k-1}$ problem for linear matrices of size $O(s^5)$ over the variable set $X_{[k]\setminus\{1\}}$. Equivalently, we need to compute the rank of these linear matrices over the skew field $U_{[k]\setminus\{1\}}$, which has $k-1$ parts in the set of partially commutative variables.

At the end, the PC-PIT$_k$ algorithm will compute a monomial m over X_1 and its coefficient, which is an ABP over the remaining variables $X_{[k]\setminus\{1\}}$. If $f \neq 0$, then given such a monomial m, as discussed in Section 2.1.1, we can efficiently find scalar matrix substitutions $\{M_x\}_{x \in X_1}$ for the X_1-variables such that the polynomial f remains nonzero. We can even ensure that the entries of each M_x is in $\{0,1\}$ and $\dim(M_x) \leqslant d+1 \leqslant s+1$ as explained in Section 2.1.1.

The PC-PIT$_k$ procedure described above computes a nonzero monomial $m \in X_1^d$ for some $d \leqslant s$ whose coefficient is a nonzero ABP $\mathcal{A}_m$ in $\mathbb{Q}\langle X_{[k]\setminus\{1\}}\rangle$. By Lemma 12, the size of $\mathcal{A}_m$ is $O(sd^2) = O(s^3)$, width $O(sd)$ and computes a polynomial of degree $\leqslant d$. Hence we can recursively apply PC-PIT$_{k-1}(\mathcal{A}_m, O(s^3), O(sd), d, X_{[k]\setminus\{1\}})$.

The PC-PIT$_{k-1}$ subroutine outputs the substitution matrices for the variables $x \in X_j : 2 \leqslant j \leqslant k$ which are of tensor product structure $I_{d_2} \otimes \cdots \otimes I_{d_{j-1}} \otimes M_x \otimes I_{d_{j+1}} \otimes \cdots \otimes I_{d_k}$ and the dimensions $d_2, \ldots, d_k$ are at most $d+1$. Combining with the substitution matrices for X_1, the final structure of the matrix substitutions for $x \in X_j$ is of the form $I_{d_1} \otimes I_{d_2} \otimes \cdots \otimes I_{d_{j-1}} \otimes M_x \otimes I_{d_{j+1}} \otimes \cdots \otimes I_{d_k}$. This follows from Observation 8. Now the theorem follows by considering the procedure above for every X_1-homogenized ABPs. $\square$

Remark 37. By Theorem 36, each $d_j \leqslant d+1$. However, we can relax the bound for each d_j and choose any larger value. This can be easily done using a standard idea of padding sufficient number of zero rows and columns to the matrix construction shown in Subsection 2.1.1. We will require this in Subsection 4.2.3, where we need to ensure that $d_j, 1 \leqslant j \leqslant k$ are distinct prime numbers bounded by $\mathrm{poly}(s)$.

4.1.1 Matrix substitution witnessing nonzero of a series

In the design of the subroutine PC-Rank, we need to find nonzero of a series over partially commutative variables. To that end, using Theorem 36 we prove the following lemma.

Lemma 38. *Let $S = b(I-L)^{-1}a$ be a series over the partially commutative variable set $X_{[k]}$. The dimension of I, L are $s \times s$, b, a are $1 \times s$ and $s \times 1$ dimensional vectors respectively. The entries in b, a, L are linear forms over $X_{[k]}$. Then, there is a deterministic polynomial time algorithm, with access to subroutine* PC-PIT$_k$ *for linear matrices, that computes matrix substitutions on which S evaluates to a nonzero matrix if $S \neq 0$.*

Proof. For $k=1$, in Section 3 we showed that finding a nonzero of the series S reduces to finding a nonzero of its s-term truncation $P_S = b\left(\sum_{k \leqslant s} L^k\right) a$ (using Lemma 18), and a scaling trick. In this section, we extend the approach for $k > 1$. Lemma 18 implies that $S = 0$ if and only if $P_S = b\left(\sum_{k \leqslant s} L^k\right) a = 0$.

Apply PC-PIT$_k$ on P_S and using Theorem 36 compute substitution matrices which has tensor product structure. For the convenience of notation, let $\underline{M}_1, \underline{M}_2, \ldots, \underline{M}_k$ be the tuples of the matrices for the variables in $X_1, X_2, \ldots, X_k$ respectively. Let t be a commutative variable and by $t\underline{M}_j$, we mean that each matrix in the tuple is scaled by the factor t. Notice that $S(t\underline{M}_1, \ldots, t\underline{M}_k)$ is nonzero since $P_S(t\underline{M}_1, \ldots, t\underline{M}_k) \neq 0$ and different t degrees homogenized components will not mix together. Let $d_1, d_2, \ldots, d_k$ be the dimension of the matrices in different components as promised by Theorem 36, and let $d = d_1 d_2 \cdots d_k$. Thus $(I-L)(t\underline{M}_1, \ldots, t\underline{M}_k)$ evaluates to a matrix of dimension sd over the variable t. Similarly, $b(t\underline{M}_1, \ldots, t\underline{M}_k)$ and $a(t\underline{M}_1, \ldots, t\underline{M}_k)$ are s dimensional vectors of matrices of dimension d. We want a value for the parameter t that makes $\det[(I-L)(t\underline{M}_1, \ldots, t\underline{M}_k)] \neq 0$ and $S = b(I-L)^{-1}a((t\underline{M}_1, \ldots, t\underline{M}_k)) \neq 0$. Hence, it suffices to avoid the roots of the univariate polynomials in t originating from the determinant computation and the entries of $b(I-L)^{-1}a[t\underline{M}_1, \ldots, t\underline{M}_k]$. Since $d = s^{O(k)}$ by Theorem 36, we can find a suitable value of t from a $\mathrm{poly}(s^k)$ size finite subset of $\mathbb{Q}$. $\square$

4.2 The procedure for PC-RANK

We are now ready to design the subroutine PC-Rank$_k$. We can write the input linear matrix T of size s as:

$$T(X_1, \ldots, X_k) = A_0 + \sum_{j=1}^{k} \sum_{x \in X_j} A_x x. \tag{7}$$

The algorithm computes the matrix substitution for each $x \in X_j$ $(1 \leqslant j \leqslant k)$ of the form

$$x \leftarrow I_{d_1} \otimes \cdots I_{d_{j-1}} \otimes M_x \otimes I_{d_{j+1}} \otimes \cdots \otimes I_{d_k}, \tag{8}$$

where matrix M_x is d_j-dimensional and the rank of the resulting scalar matrix will be $(d_1 d_2 \cdots d_k) \cdot$ pc-rank(T). Recall from the definition 24, that the substitutions of the form 8 is a type-j k-fold tensor. Consider the following definition of witness of pc-rank.

Definition 39 (Witness of pc-rank r)**.** Let $T(X_1, \ldots, X_k)$ be the given linear matrix of the form $T = A_0 + \sum_{j=1}^{k} \sum_{x \in X_j} A_x x$ such that $A_0, A_x \in \mathrm{Mat}_s(\mathbb{Q})$ for $x \in X_{[k]}$. We say that a matrix substitution of shape $\underline{d} = (d_1, \ldots, d_k)$ that assigns type-j k-fold tensor products for variables in X_j $(1 \leqslant j \leqslant k)$, is a witness of pc-rank$(T) \geqslant r$ if T evaluates to a scalar matrix of rank at least $rd_1 d_2 \cdots d_k$ after the substitution.

Now, we describe a rank increment procedure that computes new matrix assignments to the variables in X_i $(1 \leqslant i \leqslant k)$ that witness the pc-rank$(T(X_{[k]}))$ is at least $r+1$, if such a rank increment is possible. To do that, we need the following lemma and corollary as preparatory results.

Lemma 40. *Let $T = A_0 + \sum_{i=1}^{k} \sum_{x \in X_i} A_x x$ be an $s \times s$ linear matrix over variables $X_{[k]}$. For $d \in \mathbb{N}$ define $T_d = A_0 \otimes I_d + \sum_{x \in X_1} (A_x \otimes Z_x) + \sum_{i=2}^{k} \sum_{x \in X_i} (A_x \otimes I_d) x$ where $\{Z_x = (z_{x,i,j})_{1 \leqslant i,j \leqslant d}\}_{x \in X_1}$ be a set of generic matrices of noncommutative variables which are commuting with $X_2, \ldots, X_k$. Then,* pc-rank$(T_d) = d \cdot$ pc-rank(T).

Proof. Write $T_d = A_0 \otimes I_d + \sum_{x,i,j} A_{x,i,j}\, z_{x,i,j} + \sum_{i=2}^{k} \sum_{x \in X_i} (A_x \otimes I_d)\, x$ where each $\{A_{x,i,j} : x \in X_1, 1 \leqslant i, j \leqslant d\}$ is an $sd \times sd$ matrix.

Let pc-rank$(T) = r$. Then, T has a submatrix M of size r invertible over the skew field $\mathfrak{U}_{[k]}$ (by Theorem 21). By Proposition 23, there are matrix substitutions for the variables $x \in X_i : 1 \leqslant i \leqslant k$ of the form $I_{d'_1} \otimes I_{d'_2} \otimes \cdots \otimes I_{d'_{i-1}} \otimes p_x \otimes I_{d'_{i+1}} \otimes \cdots \otimes I_{d'_k}$ such that M evaluates to an invertible scalar matrix. Here, p_x is a $d'_i \times d'_i$ matrix and $d'_1, \ldots, d'_k \in \mathbb{N}$. Also, w.l.o.g, we can assume $d'_i : 1 \leqslant i \leqslant k$ to be multiple of d. In particular, let $d''_1 = d'_1 / d$.

For each $x \in X_1$, let us write the matrix p_x as a matrix of blocks of dimension $d''_1 \times d''_1$. So the $(i,j)^{th}$ block in $[d] \times [d]$ is a matrix $q_{x,i,j}$. Now, it is not hard to see that M corresponds to a submatrix of size rd in T_d which becomes invertible by the substitutions

$$z_{x,i,j} \longleftarrow q_{x,i,j} \otimes I_{d'_2} \otimes \cdots \otimes I_{d'_{i-1}} \otimes I_{d'_i} \otimes I_{d'_{i+1}} \otimes \cdots \otimes I_{d'_k},$$

for $x \in X_1$ and

$$x \longleftarrow I_{d''_1} \otimes I_{d'_2} \otimes \cdots \otimes I_{d'_{i-1}} \otimes p_x \otimes I_{d'_{i+1}} \otimes \cdots \otimes I_{d'_k},$$

for $x \in X_i : 2 \leqslant i \leqslant k$. Hence pc-rank$(T_d) \geqslant rd$.

For the other direction, as pc-rank$(T) = r$, we can write

$$T = U \cdot \left(\begin{array}{c|c} I_r & 0 \\ \hline 0 & 0 \end{array} \right) \cdot V,$$

for invertible transformations U, V over the skew field $\mathfrak{U}_{[k]}$. Hence, pc-rank$(T_d) \leqslant rd$. This proves the lemma. □

We apply Lemma 40 repeatedly to prove the following corollary.

Corollary 41. *Let $T = A_0 + \sum_{i=1}^{k} \sum_{x \in X_i} A_x x$ be an $s \times s$ linear matrix over the partially commutative set of variables $X_{[k]}$. Let $d_1, d_2, \ldots, d_k \in \mathbb{N}$, and define $T_{d_1, d_2, \ldots, d_k} = A_0 \otimes I_{d_1 d_2 \cdots d_k} + \sum_{i=1}^{k} \sum_{x \in X_i} A_x \otimes I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$ where the dimension of the generic noncommutative matrices Z_x for the variables $x \in X_i$ is d_i, and the variables in $\{Z_x\}_{x \in X_i}$ and $\{Z_x\}_{x \in X_j}$ are mutually commuting for $i \neq j$. Then,* pc-rank$(T_{d_1, d_2, \ldots, d_k}) = d_1 d_2 \cdots d_k \cdot$ pc-rank(T).

Proof. For clarity we explain the proof up to stage two where we handle the variables in X_1 and X_2. Then a simple induction on k gives the general result.

For $d_1 \in \mathbb{N}$, define $T_{d_1} = A_0 \otimes I_{d_1} + \sum_{x \in X_1} A_x \otimes Z_x + \sum_{i=2}^{k} \sum_{x \in X_i} (A_x \otimes I_{d_1})x$ where Z_x is a d_1 dimensional generic matrix. Then by Lemma 40, we know that pc-rank(T_{d_1}) = $d_1 \cdot$ pc-rank(T). Let $A'_0 = A_0 \otimes I_{d_1} + \sum_{x \in X_1} A_x \otimes Z_x$. Also, for each $x \in \bigsqcup_{i=2}^{k} X_i$, we use A'_x to denote the matrix $A_x \otimes I_{d_1}$. Thus,

$$T_{d_1} = A'_0 + \sum_{i=2}^{k} \sum_{x \in X_i} A'_x x.$$

Now replace the variables $x \in X_2$ by generic matrices Z_x of dimension d_2 to get the matrix

$$T_{d_1,d_2} = A'_0 \otimes I_{d_2} + \sum_{i=3}^{k} \sum_{x \in X_i} (A'_x \otimes I_{d_2})x + \sum_{x \in X_2} A'_x \otimes Z_x.$$

Applying the Lemma 40 again, we know that pc-rank(T_{d_1,d_2}) = $d_2 \cdot$ pc-rank(T_{d_1}) = $d_1 d_2 \cdot$ pc-rank(T). Note that to get T_{d_1,d_2} from T, we need to substitute the variables $x \in X_1$ by matrices of the form $Z_x \otimes I_{d_2}$. Similarly, the matrices for $x \in X_2$ are given by $I_{d_1} \otimes Z_x$. Repeating the process k times we get the desired result. □

4.2.1 Rank increment step

We now return to the construction of the subroutine PC-Rank$_k$. The main idea is that, given an input linear matrix T over $X_{[k]}$, we do an induction on the rank parameter r. Clearly for the base case ($r = 1$), we can easily make a linear form nonzero after the evaluation. Suppose that we have already computed a rank r witness $\underline{M}$, which is a type-j k-fold tensor product matrix assignments for the variables in X_j ($1 \leqslant j \leqslant k$) such that:

- rank($T(\underline{M})$) is at least rd' where $(d_1, d_2, \ldots, d_k)$ is the shape of the tensor and $d' = d_1 d_2 \cdots d_k$.
- Moreover, for each $1 \leqslant j \leqslant k$, $d_j \leqslant s^3$ and $d_1, \ldots, d_k$ are distinct prime numbers.

Let $T_{d'}(Z)$ denote the matrix obtained from T by replacing the variables $x \in \bigsqcup_{i=1}^{k} X_i$ by the matrices $I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$ where the dimension of the generic matrix Z_x is d_i. By Corollary 41, pc-rank($T_{d'}(Z)$) = $d' \cdot$ pc-rank(T). Let Z denote the tuple $(Z_1, \ldots, Z_k)$ where $Z_\ell = \{Z_x\}_{x \in X_\ell}$ for each ℓ. Equivalently, if we regard each matrix Z_ℓ as the set of variables $\{z_{x,i',j'}\}_{1 \leqslant i',j' \leqslant d_\ell; x \in X_\ell}$, then Z is essentially the new set of partially commutative variables. That is, in each Z_ℓ the variables are noncommuting and variables across different sets Z_{ℓ_1}, Z_{ℓ_2}, for $\ell_1 \neq \ell_2$, are mutually commuting.

Next, in $T_{d'}(Z)$ replace each matrix $Z_x \otimes I_{d_2} \otimes \cdots \otimes I_{d_k}$ for $x \in X_1$ by

$$(Z_x + M_x) \otimes I_{d_2} \otimes \cdots \otimes I_{d_k}.$$

Similarly, the matrices $I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}$ corresponding to $x \in X_{[k]\setminus\{1\}}$ are replaced by

$$I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes (Z_x + M_x) \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}.$$

For the simplicity, we write the matrix obtained as $T_{d'}(Z + \underline{M})$.

Notice that,

$$T_{d'}(Z+\underline{M}) = T_1'' + \sum_{i=1}^{k}\sum_{x\in X_i} A_x \otimes I_{d_1} \otimes \cdots \otimes I_{d_{i-1}} \otimes Z_x \otimes I_{d_{i+1}} \otimes \cdots \otimes I_{d_k}, \tag{9}$$

recalling the discussion in Section 1.1 (see Equation 3).

Since the rank of a linear matrix is invariant under shifting of the variables by scalars (See, footnote 6), we get that pc-rank$(T_{d'}(Z+\underline{M}))$ = pc-rank$(T_{d'}(Z))$.

For invertible transformations U, V over $\mathbb{Q}$, we can write

$$T_{d'}(Z+\underline{M}) = U\left(\begin{array}{c|c} I_{rd'} - L & A \\ \hline B & C \end{array}\right)V.$$

Furthermore,

$$T_{d'}(Z+\underline{M}) = UU'\left(\begin{array}{c|c} I_{rd'} - L & 0 \\ \hline 0 & C - B(I_{rd'}-L)^{-1}A \end{array}\right)V'V. \tag{10}$$

$$\text{Here, } U' = \left(\begin{array}{c|c} I_{rd'} & 0 \\ \hline B(I_{rd'}-L)^{-1} & I_{(s-r)d'} \end{array}\right), \qquad V' = \left(\begin{array}{c|c} I_{rd'} & (I_{rd'}-L)^{-1}A \\ \hline 0 & I_{(s-r)d'} \end{array}\right).$$

Notice that L, A, B, C are linear matrices over the variables in $Z_1, Z_2, \ldots, Z_k$. The $(\ell_1, \ell_2)^{th}$ entry of $C - B(I_{rd'}-L)^{-1}A$ is given by $C_{\ell_1\ell_2} - B_{\ell_1}(I_{rd'}-L)^{-1}A_{\ell_2}$ where B_{ℓ_1} is the ℓ_1^{th} row vector of B and A_{ℓ_2} is the ℓ_2^{th} column vector of A. We prove the following lemma which is the partially commutative version of Lemma 29.

Lemma 42. pc-rank$(T) > r$ *if and only if* $S_{\ell_1\ell_2} = C_{\ell_1\ell_2} - B_{\ell_1}(I_{rd'}-L)^{-1}A_{\ell_2} \neq 0$ *for some choice of* ℓ_1, ℓ_2.

Proof. Suppose pc-rank$(T) > r$. Then, by Corollary 41, pc-rank$(T_{d'}(Z+\underline{M}))$ = pc-rank$(T_{d'}(Z)) > rd'$. However, if $C - B(I_{rd'}-L)^{-1}A$ is a zero matrix, this is impossible.

Conversely, if $(T_{d'}(Z+\underline{M}))_{\ell_1\ell_2} = C_{\ell_1\ell_2} - B_{\ell_1}(I_{rd'}-L)^{-1}A_{\ell_2}$ is nonzero for some indices ℓ_1, ℓ_2, we can find (partially commutative) matrix substitutions to the variables in Z such that $T_{d'}(Z+\underline{M})$ evaluated on such substitutions (let say of dimension $\hat{d}$) will be of rank more than $rd'\hat{d}$. Then pc-rank$(T_{d'}(Z+\underline{M})) > rd'$ implying that pc-rank$(T) > r$ by Corollary 41. ◻

4.2.2 A partially commutative ABP identity testing reduction step

Now we vary over all choices for ℓ_1, ℓ_2 and apply Lemma 38 to find a nonzero of the series represented by $S_{\ell_1\ell_2}$ for some choice of ℓ_1, ℓ_2.[8] Next, we describe how to update the matrix tuple $\underline{M}$ to a new tuple that will be the assignment for the $X_{[k]}$ variables.

Suppose, by Lemma 38, we obtain matrix assignments $\{M'_{x,i,j}\}_{x\in X_1, 1\leqslant i,j\leqslant d_\ell}$ to the Z_ℓ variables, $1 \leqslant \ell \leqslant k$. Consider $x \in X_\ell$ $(1 \leqslant \ell \leqslant k)$. The matrix assignment for Z_x will be the matrix M'_x obtained by replacing the variable $z_{x,i,j} : 1 \leqslant i, j \leqslant \ell$ by the matrix $M'_{x,i,j}$ of dimension p_ℓ. Now, let

$$M''_x = M_x \otimes I_{p_\ell} + M'_x.$$

For $x \in X_\ell$, we substitute x by type-ℓ k-fold tensor

$$I_{d''_1} \otimes \cdots \otimes I_{d''_{\ell-1}} \otimes M''_x \otimes I_{d''_{\ell+1}} \otimes \cdots \otimes I_{d''_k},$$

[8] Notice that $C_{\ell_1\ell_2}$ is a linear term and the degree of the other terms in the series is at least 2.

where we note that each $M''_x, x \in X_\ell$ is of dimension $d''_\ell = d_\ell p_\ell$. We denote the resulting tuple of matrices by $\widetilde{\underline{M}}$. Proof of the next claim is analogous to the proof of Corollary 31.

Claim 43. $\operatorname{rank}(T(\widetilde{\underline{M}})) > r d''_1 d''_2 \cdots d''_k$.

Remark 44. We observe the following additional properties of our construction:

1. W.l.o.g, we can ensure that $p_1, p_2, \ldots, p_k$ are distinct odd primes such that each $p_\ell > d_j$ as discussed in Remark 37.
2. The above choice of the primes p_ℓ ensures that the dimensions $d''_\ell, 1 \leqslant \ell \leqslant k$ are pairwise relatively prime since the d_ℓ are distinct prime numbers.

4.2.3 Rounding step

Recall from the last section, we have already computed a matrix tuple $\widetilde{\underline{M}}$ of shape $(d''_1, \ldots, d''_k)$ such that $\operatorname{rank}(T(\widetilde{\underline{M}})) > r d''_1 \cdots d''_k$, where the d''_ℓ are all pairwise relatively prime. We now describe the algorithm to obtain a witness of pc-rank $r+1$ if $\operatorname{rank}(T) \geqslant r+1$.

Lemma 45. *Given a linear matrix T over $X_{[k]}$ of size s and matrix tuple $\widetilde{\underline{M}}$ of shape $(d''_1, \ldots, d''_k)$ such that* $\operatorname{rank}(T(\widetilde{\underline{M}})) > r d''_1 d''_2 \cdots d''_k$ *and the d''_i are pairwise relatively prime, we can compute another matrix tuple $\widehat{\underline{M}}$ in deterministic* $\operatorname{poly}(s, d''_1, \ldots, d''_k)$ *time such that* $\operatorname{rank}(T(\widehat{\underline{M}})) \geqslant (r+1) \cdot d''_1 d''_2 \cdots d''_k$.

Proof. If $\operatorname{rank}(T(\widetilde{\underline{M}}))$ is a multiple of each d''_i then the hypothesis already implies $\operatorname{rank}(T(\widetilde{\underline{M}})) \geqslant (r+1) \cdot d''_1 d''_2 \cdots d''_k$, and there is nothing to prove. Now, suppose $\operatorname{rank}(T(\widetilde{\underline{M}}))$ is not a multiple of d''_i for some $i \in [k]$. The idea is to find a $d''_i \times d''_i$ matrix substitution M''_x for each $x \in X_i$ and update the i^{th} component of the matrix tuple $\widetilde{\underline{M}}$ such that $\operatorname{rank}(T(\underline{M}''))$ is a multiple of d''_i where $\underline{M}''$ is the updated matrix tuple. To do so, we first substitute each $x \in X_{[k]\setminus\{i\}}$ by the restriction of the matrix tuple $\widetilde{\underline{M}}$ of shape $(d''_1, \ldots, d''_{i-1}, d''_{i+1}, \ldots, d''_k)$ by dropping the i^{th} component and obtain a linear matrix $T_{[k]\setminus\{i\}}(X_i)$.

Now, we are left with an instance of the noncommutative rank computation over X_i variables. By Lemma 26, we can find matrix substitutions $M''_x : x \in X_i$ such that $\operatorname{rank}(T_{[k]\setminus\{i\}}(\{M''_x\}))$ is a multiple of d''_i. It also updates the matrix tuple $\widetilde{\underline{M}}$ to $\underline{M}''$ by updating only the i^{th} component of $\widetilde{\underline{M}}$ to M''_x. Now $\operatorname{rank}(T(\underline{M}'')) > r d''_1 \cdots d''_k$ and $\operatorname{rank}(T(\underline{M}''))$ is a multiple of d''_i.

We now do this for each $j \in [k] \setminus \{i\}$, to find a matrix substitution $\widehat{\underline{M}}$ such that $\operatorname{rank}(T(\widehat{\underline{M}}))$ is a multiple of d''_j for each $j \in [k]$. As the d''_j are pairwise relatively prime, $\operatorname{rank}(T(\widehat{\underline{M}}))$ is also a multiple of $d''_1 d''_2 \cdots d''_k$. Moreover, $\operatorname{rank}(T(\widehat{\underline{M}})) > r d''_1 \cdots d''_k$. Therefore, $\operatorname{rank}(T(\widehat{\underline{M}})) \geqslant (r+1) d''_1 \cdots d''_k$. ☐

Next, we describe the blow-up control step.

4.2.4 Blow-up and shape control step

Now the plan is to find another rank $r+1$ witness such that the dimension of the i^{th} component is bounded by s^3. Moreover, the witness is of *prime shape* $(p_1, \ldots, p_k)$ where the p_i are distinct prime numbers. We need the following result about primes in short intervals, along with a nontrivial generalization of Lemma 32, to prove the next lemma.

Theorem 46 (prime number theorem in short interval [LY92]). *Let n be a sufficiently large number, and $\pi(n)$ be the number of primes $\leqslant n$. Moreover, let $n' = n^{\theta}$ for $1/2 \leqslant \theta \leqslant 7/12$. Then,*

$$1.01\frac{n'}{\log n} \geqslant \pi(n) - \pi(n-n') \geqslant 0.99\frac{n'}{\log n}$$

We are now ready to present the blow-up control step. For our purpose, we will choose $\theta = 0.6$.

Lemma 47. *Suppose T is a linear matrix of size s over $X_{[k]}$ and $\widehat{\underline{M}}$ is a matrix tuple of shape $(d''_1, \ldots, d''_k)$ such that*

- $\mathrm{rank}(T(\widehat{\underline{M}})) \geqslant (r+1)d''_1 d''_2 \cdots d''_k$.
- *The dimensions $d''_i, 1 \leqslant i \leqslant k$ are pairwise relatively prime. Moreover, each d''_i is a product of two distinct odd primes.*

Then for all but finitely many s, in deterministic $\mathrm{poly}(s, d''_1, \ldots, d''_k)$ time, we can compute another matrix tuple $\widehat{\underline{N}}$ of prime shape $(p_1, \ldots, p_k)$ such that $\mathrm{rank}(T(\widehat{\underline{N}})) \geqslant (r+1)p_1 \cdots p_k$ and for each i, $p_i \leqslant s^3$ is a prime number.

Proof. We will prove the statement by induction, replacing d''_i by prime p_i for increasing indices i. Inductively assume that we have computed a matrix tuple $\widetilde{\underline{M}}$ of shape $(p_1, \ldots, p_\ell, d''_{\ell+1}, \ldots, d''_k)$ such that

- Each $p_j \leqslant s^3$ is an odd prime.
- $\mathrm{rank}(T(\widetilde{\underline{M}})) \geqslant (r+1)p_1 p_2 \cdots p_\ell d''_{\ell+1} \cdots d''_k$.
- The dimensions $p_1, p_2, \ldots, p_\ell$ are distinct primes that are also relatively prime to each d''_i, $i > \ell$.

Notice that the base case is $\ell = 0$. In the inductive step, our goal is to replace $d''_{\ell+1}$ by a prime $p_{\ell+1}$ satisfying the above. That will complete the proof.

Consider an invertible sub-matrix A in T of size $r+1$. We can find such A since the rank of $T(\widetilde{\underline{M}})$ is $\geqslant (r+1)p_1 p_2 \cdots p_\ell d''_{\ell+1} \cdots d''_k$.

The following claim summarize how we will be applying the number-theoretic Theorem 46 to find the prime $p_{\ell+1}$.

Claim 48. *For all but finitely many d (depending on k) there are at least $2k+1$ many prime numbers in the interval $(d - d^{0.6}, d]$.*

We will apply the claim to $d = d''_{\ell+1}$. We can assume without loss of generality that $d''_{\ell+1} > s^3$. This is because we can always double the dimension $d''_{\ell+1}$ by making the matrix components corresponding to $X_{\ell+1}$ in $\widetilde{\underline{M}}$ block diagonal with two blocks. Notice that the resulting matrix tuple is still a witness of rank at least $r+1$. Furthermore, notice that all the primes p_j and the dimensions d''_ι, $\iota > \ell$ are all still pairwise relatively prime as $d''_{\ell+1}$ only changed by factors of 2.

By abuse of notation, let $\widetilde{\underline{M}}$ still denote the modified matrix tuple with $d''_{\ell+1} > s^3$. By the above Claim 48, for all but finitely many s, we can find a prime p in the interval $(d''_{\ell+1} - d''^{0.6}_{\ell+1}, d''_{\ell+1}]$ that is relatively prime to all the p_j and to each d''_ι, $\iota > \ell + 1$.

Now from the matrices in $\widetilde{\underline{M}}$, remove the last $d''_{\ell+1} - p$ many rows and columns of the first component matrices (namely, the substitutions for variables in $X_{\ell+1}$) and keep other substitutions as they are. This yields another tuple $\underline{M}'$ of matrix substitutions for the $X_{[k]}$ variables.

Let $\Delta = \prod_{j=1}^{\ell} p_j \prod_{\iota>\ell+1} d''_\iota$. We claim that $\text{rank}(A(\underline{M}')) > rp\Delta$. Suppose not. Then, we have:

$$\begin{aligned}
\text{rank}(A(\widehat{\underline{M}})) &\leqslant \text{rank}(A(\underline{M}')) + 2(r+1)(d''_{\ell+1} - p)\Delta \\
&\leqslant rp\Delta + 2(r+1)(d''_1 - p)\Delta \\
&= \Delta(rp + 2(r+1)(d''_{\ell+1} - p)) \\
&= \Delta(2(r+1)d''_{\ell+1} - (r+2)p) \\
&< (r+1)d''_{\ell+1}\Delta
\end{aligned}$$

which contradicts the inductive assumption. To see the last strict inequality we observe that $(r+1)d''_{\ell+1} < (r+2)p$. This follows from the following:

$$(r+1)(d''_{\ell+1} - p) \leqslant (r+1)d''^{0.6}_{\ell+1} \leqslant (s+1)d''^{0.6}_{\ell+1} \leqslant d''^{0.95}_{\ell+1} < p$$

because $(s+1) < d''^{0.34}_{\ell+1}$ and $p > d''_{\ell+1} - d''^{0.6}_{\ell+1}$.

We can now apply Lemma 45, with the matrix tuple $\underline{M}'$ as input, to find a new matrix tuple on which T will evaluate to a matrix of rank $\geqslant (r+1)p\Delta$. Now, if $p > s^3$ we can repeat the above process with p instead of $d''_{\ell+1}$ until we finally get $p \leqslant s^3$. Then we set $p_{\ell+1} = p$ completing the inductive step of the proof.

To summarize, we will finally obtain the claimed matrix substitution $\widehat{\underline{N}}$ of prime shape $(p_1, \ldots, p_k)$ where each $p_i \leqslant s^3$. The runtime bound is easy to verify. ◻

Remark 49. The dimension of the final $(r+1)$-rank witness $\widehat{\underline{N}}$ is bounded by s^{3k}.

4.2.5 Pseudo-code for rank increment

Given an input matrix T over $X_{[k]}$ and type-j k-fold tensor product matrix assignments for the variables in X_j ($1 \leqslant j \leqslant k$) such that $\text{rank}(T(\underline{M}))$ is at least rd' where $\underline{d} = (d_1, d_2, \ldots, d_k)$ is the shape of the tensor and $d' = d_1 d_2 \cdots d_k$, we describe the pseudo-code of the rank increment procedure described above that finds another set of assignments to the variables in X_j ($1 \leqslant j \leqslant k$) that witness the pc-rank($T(X_{[k]})$) is at least $r+1$, if such a rank increment is possible. Moreover, for each $1 \leqslant j \leqslant k : d_j \leqslant s^3$ and $\underline{M}$ represents the entire tuple of matrix assignment.

Algorithm for Rank-Increment $(T_{[k]}, \underline{M}, r)$

Input : A linear matrix T over $X_{[k]}$ and the matrix tuple $\underline{M}$ such that $\text{rank}(T(\underline{M}))$ is at least rd' where the shape of the matrix tuples in $\underline{M}$ are given by $\underline{d} = (d_1, d_2, \ldots, d_k)$ such that the $d_j \leqslant s^3$ ($1 \leqslant j \leqslant k$) are distinct prime numbers and $d' = d_1 d_2 \cdots d_k$.

Output : Find another set of matrix assignments of shape $\underline{d} = (d_1, \ldots, d_k)$ for $x \in X_{[k]}$ that witness the pc-rank(T) $\geqslant r+1$, if such a rank increment is possible and the $d_j \leqslant s^3$ are distinct prime numbers.

Steps : 1. Using the Z variables and the matrix shift, construct the linear matrix $T_{d'}(Z + \underline{M})$ as shown in Equation 9.

2. Using Gaussian elimination, convert the matrix $T_{d'}(Z + \underline{M})$ to the block diagonal shape shown in Equation 10.
3. Use Lemma 38, to find the nonzero of a series originating from the bottom right of the block. If it fails to find a nonzero STOP the procedure.
4. Use Claim 43 to compute a new set of matrix assignments of dimension $d''_1, d''_2, \ldots, d''_k$ (the d''_j are pairwise relatively prime) to the variables in $X_1, X_2, \ldots, X_k$, such that after the evaluation, the rank of the resulting matrix is strictly more than $r \cdot d''_1 d''_2 \cdots d''_k$.
5. Use Lemma 45 and Lemma 47 to implement the rounding and the blow-up control steps and compute the matrix assignments that witness pc-rank$(T) \geqslant r + 1$. Moreover, the dimension of each component of the witness is bounded by s^3.

We complete the section with the proof of the main theorems. For the convenience of the reader, we restate the theorems.

Theorem 50 (Restate of Theorem 1)**.** *Given an $s \times s$ matrix T whose entries are $\mathbb{Q}$-linear forms over the partially commutative set of variables $X_{[k]}$ (where $|X_i| \leqslant n$ for $1 \leqslant i \leqslant k$ and w.l.o.g $n \leqslant s$), the rank of T over $\mathfrak{U}_{[k]}$ can be computed in deterministic $s^{2^{O(k \log k)}}$ time. The bit complexity of the algorithm is also bounded by $s^{2^{O(k \log k)}}$.*

Proof. Firstly note that, due the blow-up control step, the shape of the matrix tuples is always determined by the size of the input matrix thus it remains as $\underline{d} = (d_1, d_2, \ldots, d_k)$ where each $d_i \leqslant s^3$. Also, since the pc-rank(T) is bounded by s, the subroutine Rank-Increment can be called for at most s times. Let $t_k(s)$ be the time taken by the procedure Rank-Increment from rank r to rank $r+1$. The size of the matrix $T_{d'}(Z + \underline{M})$ is at most $sd' = s^{O(k)}$ since $d' = d_1 d_2 \cdots d_k$. Hence Step 1 and Step 2 can be performed in $s^{O(k)}$ time. In Step 3, the application of Lemma 38 calls PC-PIT$_k$ on a linear matrix of size $s^{O(k)}$ and additional $s^{O(k)}$ time for linear algebraic computation. As shown in Lemma 45 and Lemma 47 that Step 5 takes at most $s^{O(k)}$ time.

Let $T_1(s, k)$ be the running time of the PC-PIT$_k$ subroutine on an ABP of size s over $X_{[k]}$, and $T_2(s, k)$ be the running time of the PC-Rank$_k$ subroutine on a linear matrix of size s over $X_{[k]}$. Then, for a suitable constant $\beta > 0$ we can bound

$$t_k(s) \leqslant s^{\beta k} T_1(s^{\beta k}, k) + s^{\beta k}.$$

Now, we simultaneously analyze the recurrences for $T_1(s, k)$ and $T_2(s, k)$. Notice that, $T_1(s, k) \leqslant T_2(O(s), k)$, since size s ABPs have linear pencils of size $O(s)$ (Proposition 7). From Theorem 36 and from the time analysis of Rank-Increment subroutine as shown above, as $T_2(s, k) \leqslant s t_k(s)$ we have:

$$T_1(s, k) \leqslant s T_1(s^4, k-1) + s^6 T_2(s^6, k-1) + s^{O(1)}$$
$$T_2(s, k) \leqslant s T_1(s^{\gamma k}, k) + s^{\gamma k}.$$

for some constant $\gamma > 0$. From the first inequality above, $T_1(s, k) \leqslant 2^k s^{O(1)} T_2(s^6, k-1)$ for all but finitely many s. Combined with the second inequality above, we have $T_2(s, k) \leqslant s^{\tau k} T_2(s^{\tau k}, k-1)$ for a suitable constant $\tau > \beta$.

$$\text{Therefore,} \quad T_2(s, k) \leqslant s^{\tau k} T_2(s^{\tau k}, k-1) \leqslant s^{\tau k} \cdot s^{\tau k} \cdots s^{\tau k} \cdot T_{\textsc{NSingular}}(s^{(\tau k)^k}),$$

where $T_2(s, 1) = T_{\text{NSingular}}(s) = \text{poly}(s)$ is the running time of the NSingular algorithm on a linear matrix of size s. Therefore, we have

$$T_2(s, k) \leqslant (s^{(\tau k)^2}) \,\text{poly}(s^{(\tau k)^k}) \leqslant s^{2^{O(k \log k)}}.$$

We can bound the bit complexity of the algorithm along the same line and noting the fact that the bit complexity of the NSingular algorithm is polynomially bounded. □

Next, we prove Theorem 3.

Theorem 51 (Restate of Theorem 3). *Given an ABP of size s whose edges are labeled by $\mathbb{Q}$-linear forms over the partially commutative set of variables $X_{[k]}$ (where $|X_i| \leqslant n \leqslant s$ (w.l.o.g) for $1 \leqslant i \leqslant k$), there is a deterministic $s^{2^{O(k \log k)}}$ time algorithm to check whether the ABP computes the zero polynomial. As a corollary, the equivalence testing of k-tape weighted automata can be solved in deterministic polynomial time for $k = O(1)$. The bit complexity of the algorithm is also bounded by $s^{2^{O(k \log k)}}$.*

Proof. The proof follows directly from the analysis of the recurrence for $T_2(s, k)$ in the proof of Theorem 50 above. □

5 Discussion

We find the interplay between symbolic determinant identity testing, concepts from formal language theory, and noncommutative algebra very fascinating. Apart from yielding a deterministic polynomial-time algorithm for the k-tape weighted automata equivalence problem, the most interesting aspect of the PC-Singular problem is that it provides a common framework spanning both Singular and NSingular. We state a few questions for further study.

1. It would be satisfactory to obtain a deterministic algorithm for PC-Singular over a k-partitioned set of n variables such that setting $k = 1$ captures the best-known algorithm for NSingular and setting $k = n$ yields the best-known algorithm for Singular. For $k = 1$, we obtain a deterministic polynomial-time algorithm for NSingular. In contrast, as the runtime of our algorithm is doubly exponential in k, applied to the Singular problem (where $k = n$) the time bound becomes even worse than an exhaustive search. Of course, finding an $(nsk)^{O(1)}$ algorithm for PC-Singular would be a breakthrough as it would imply a circuit lower bound [KI04].

2. It is to be noted that the running time of the randomized algorithm for equivalence testing of k-tape weighted automata by Worrell [Wor13] is indeed $(ns)^{O(k)}$. Thus, it would be plausible and interesting to obtain a *deterministic* algorithm for equivalence testing of k-tape weighted automata with runtime closer to $(ns)^{O(k)}$.

3. Another interesting problem is to understand the complexity of the equivalence testing of multi-tape weighted automata for unbounded number of tapes.

References

[ACDM21] Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. Equivalence testing of weighted automata over partially commutative monoids. In

Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 10:1–10:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[ACG+22] Vikraman Arvind, Abhranil Chatterjee, Utsab Ghosal, Partha Mukhopadhyay, and C. Ramya. On identity testing and noncommutative rank computation over the free skew field. *CoRR*, abs/2209.04797, 2022.

[ACG+23] Vikraman Arvind, Abhranil Chatterjee, Utsab Ghosal, Partha Mukhopadhyay, and C. Ramya. On identity testing and noncommutative rank computation over the free skew field. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 6:1–6:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[AL50] A. S. Amitsur and J. Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950.

[Ami55] S.A Amitsur. The T-ideals of the free rings. *J. of London Math. Soc.*, 20:470–475, 1955.

[Ami66] S.A Amitsur. Rational identities and applications to algebra and geometry. *Journal of Algebra*, 3(3):304 – 359, 1966.

[AMS10] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on noncommutative and commutative polynomial identity testing. *Computational Complexity*, 19(4):521–558, 2010.

[BBJP19] Vishwas Bhargava, Markus Bläser, Gorav Jindal, and Anurag Pandey. A deterministic PTAS for the algebraic rank of bounded degree polynomials. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 647–661. SIAM, 2019.

[Bee76] C. Beeri. An improvement on valiant's decision procedure for equivalence of deterministic finite turn pushdown machines. *Theoretical Computer Science*, 3(3):305 – 320, 1976.

[BFG+19] Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Towards a theory of non-commutative optimization: Geodesic 1st and 2nd order methods for moment maps and polytopes. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 845–861. IEEE Computer Society, 2019.

[Bir73] Malcolm Bird. The equivalence problem for deterministic two-tape automata. *J. Comput. Syst. Sci.*, 7(2):218–236, 1973.

[BJP18] Markus Bläser, Gorav Jindal, and Anurag Pandey. A deterministic PTAS for the commutative rank of matrix spaces. *Theory Comput.*, 14(1):1–21, 2018.

[CF69] P. Cartier and D. Foata. Problémes combinatoires de commutation et réarrangements. *Lecture Notes in Mathematics*, 1969.

[Coh71] P. M. Cohn. The Embedding of Firs in Skew Fields. *Proceedings of the London Mathematical Society*, s3-23(2):193–213, 10 1971.

[Coh95] P. M. Cohn. Skew fields: Theory of general division rings. In *Encyclopedia of Mathematics and its Applications 57*, 1995.

[DK21] Manfred Droste and Dietrich Kuske. Weighted automata. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 113–150. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978.

[DM97] Volker Diekert and Yves Métivier. *Partial Commutation and Traces*, pages 457–533. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[DM17] Harm Derksen and Visu Makam. Polynomial degree bounds for matrix semi-invariants. *Advances in Mathematics*, 310:44–63, 2017.

[DM20] Harm Derksen and Visu Makam. Algorithms for orbit closure separation for invariants and semi-invariants of matrices. *Algebra & Number Theory*, 14(10):2791–2813, 2020.

[Edm67] Jack Edmonds. System of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sets.*, B 71:241–245, 1967.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines (Vol A)*. Pure and Applied Mathematics. Academic Press, 1974.

[FG82] Emily P. Friedman and Sheila A. Greibach. A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors. *SIAM J. Comput.*, 11:166–183, 1982.

[FR04] Marc Fortin and Christophe Reutenauer. Commutative/noncommutative rank of linear matrices and subspaces of matrices of low rank. *Séminaire Lotharingien de Combinatoire [electronic only]*, 52, 01 2004.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013.

[GGdOW16] Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. 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 109–117. IEEE Computer Society, 2016.

[Gri68] T. V. Griffiths. The unsolvability of the equivalence problem for nondeterministic generalized machines. *J. ACM*, 15(3):409–413, July 1968.

[GS20] Hayk A. Grigoryan and Samvel K. Shoukourian. Polynomial algorithm for equivalence problem of deterministic multitape finite automata. *Theor. Comput. Sci.*, 833:120–132, 2020.

[HH21] Masaki Hamada and Hiroshi Hirai. Computing the nc-rank via discrete convex optimization on CAT(0) spaces. *SIAM J. Appl. Algebra Geom.*, 5(3):455–478, 2021.

[HK91] Tero Harju and Juhani Karhumäki. The equivalence problem of multitape finite automata. *Theor. Comput. Sci.*, 78(2):347–355, 1991.

[HW15] Pavel Hrubeš and Avi Wigderson. Non-commutative arithmetic circuits with division. *Theory of Computing*, 11(14):357–393, 2015.

[IKQS15] Gábor Ivanyos, Marek Karpinski, Youming Qiao, and Miklos Santha. Generalized wong sequences and their applications to edmonds' problems. *J. Comput. Syst. Sci.*, 81(7):1373–1386, 2015.

[IMQ22] Gábor Ivanyos, Tushant Mittal, and Youming Qiao. Symbolic determinant identity testing and non-commutative ranks of matrix lie algebras. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 87:1–87:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[IQ19] Gábor Ivanyos and Youming Qiao. Algorithms based on *-algebras, and their applications to isomorphism of polynomials with one secret, group isomorphism, and polynomial identity testing. *SIAM J. Comput.*, 48(3):926–963, 2019.

[IQS17] Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Non-commutative edmonds' problem and matrix semi-invariants. *Comput. Complex.*, 26(3):717–763, 2017.

[IQS18] Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *Computational Complexity*, 27(4):561–593, Dec 2018.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004.

[KVV20] Igor Klep, Victor Vinnikov, and Jurij Volčič. Multipartite rational functions. *Documenta Math.*, 25:1285–1313, 2020.

[Lam01] T.Y. Lam. *A First Course in Noncommutative Rings (Second Edition)*. Graduate Texts in Mathematics. Springer, 2001.

[Lov89] László Lovász. Singular spaces of matrices and thier application in combinatorics. *Bulletin of Brazilian Mathematical Society*, 20(1):87–99, 1989.

[LS12] Sylvain Lombardy and Jacques Sakarovitch. The removal of weighted ϵ-transitions. In Nelma Moreira and Rogério Reis, editors, *Implementation and Application of Automata*, pages 345–352, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[LY92] Shituo Lou and Qi Yao. A chebychev's type of prime number theorem in a short interval ii. *Hardy-Ramanujan Journal*, 15:1–33, 1992.

[Mal37] A. Malcev. On the immersion of an algebraic ring into a field. *Mathematische Annalen*, 113:DCLXXXVI–DCXCI, 1937.

[Maz95] Antoni W. Mazurkiewicz. Introduction to trace theory. In Volker Diekert and Grzegorz Rozenberg, editors, *The Book of Traces*, pages 3–41. World Scientific, 1995.

[Pie82] Richard S. Pierce. *Associative Algebras*. Springer-Verlag, 1982.

[RS59] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RW19] Orit E. Raz and Avi Wigderson. Subspace arrangements, graph rigidity and derandomization through submodular optimization. *CoRR*, abs/1901.09423, 2019.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.

[SY10] 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.

[Val74] Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123 – 133, 1974.

[Wor13] James Worrell. Revisiting the equivalence problem for finite multitape automata. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 422–433, 2013.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.

A Appendix

The idea is to reduce the computation of pc-rank of a matrix with $\mathfrak{U}_{[k]}$ entries to pc-rank computation of a linear matrix incurring a small blow-up in the size. To show the reduction, we need the following lemma.

Lemma 52. *Let $X = X_{[k]}$ and $\mathfrak{U}_{[k]}$ be the universal skew field over $\mathbb{F}\langle X_{[k]}\rangle$. Let $P \in {\mathfrak{U}_{[k]}}^{m\times m}$ such that,*

$$P = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where $A \in \mathfrak{U}_{[k]}^{r\times r}$ is invertible. Then,

$$\text{pc-rank}(P) = r + \text{pc-rank}(D - CA^{-1}B),$$

Proof. If Q is an $m \times m$ invertible matrix over $\mathfrak{U}$ then

$$\text{pc-rank}(QP) = \text{pc-rank}(PQ) = \text{pc-rank}(P).$$

For if $P = MN$ then $QP = (QM)N$ and if $QP = MN$ then $P = (Q^{-1}M)N$. Similarly for PQ.

The matrix

$$\begin{bmatrix} A^{-1} & 0 \\ 0 & I_{m-r} \end{bmatrix}$$

is full rank. Similarly, the matrix

$$\begin{bmatrix} I_r & 0 \\ -C & I_{m-r} \end{bmatrix}$$

is full rank because

$$\begin{bmatrix} I_r & 0 \\ -C & I_{m-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ C & I_{m-r} \end{bmatrix} = \begin{bmatrix} I_r & 0 \\ 0 & I_{m-r} \end{bmatrix}.$$

Hence, pc-rank(P) equals pc-rank(R) where

$$R = \begin{bmatrix} I_r & 0 \\ -C & I_{m-r} \end{bmatrix} \cdot \begin{bmatrix} A^{-1} & 0 \\ 0 & I_{m-r} \end{bmatrix} \cdot \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I_r & A^{-1}B \\ 0 & D - CA^{-1}B \end{bmatrix}$$

Post-multiplying by the invertible matrix $\begin{bmatrix} I_r & -A^{-1}B \\ 0 & I_{m-r} \end{bmatrix}$ we obtain $\begin{bmatrix} I_r & 0 \\ 0 & D - CA^{-1}B \end{bmatrix}$.

It is easy to see that its inner rank is r + pc-rank$(D - CA^{-1}B)$. □

For the sake of reading, we restate Lemma 35.

Lemma 53 (Restate of Lemma 35)**.** *Let $X = X_{[k]}$ be a set of partially commutative variables. Let $M \in \mathbb{F}\langle X_{[k]}\rangle^{m\times m}$ be a matrix where each $(i,j)^{th}$ entry M_{ij} is computed as the $(1,s)^{th}$ entry of the inverse of a linear pencil L_{ij} of size s. Then, one can construct a linear pencil L of size $m^2 s + m$ such that,*

$$\text{pc-rank}(L) = m^2 s + \text{pc-rank}(M).$$

Proof. We first describe the construction of the linear pencil L and then argue the correctness.

$$\text{Let} \quad L = \left[\begin{array}{cccc|c} L_{11} & 0 & \cdots & 0 & B_{11} \\ 0 & L_{12} & \cdots & 0 & B_{12} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & L_{mm} & B_{mm} \\ \hline -C_{11} & -C_{12} & \cdots & -C_{mm} & 0 \end{array}\right], \tag{11}$$

where each C_{ij} is an $m \times s$ and B_{ij} is an $s \times m$ rectangular matrix defined below. Let e_i denote the column vector with 1 in the i^{th} entry and the remaining entries are zero. We define

$$C_{ij} = \left[\begin{array}{c|c|c|c} e_i & 0 & \cdots & 0 \end{array}\right] \quad \text{and,} \quad B_{ij} = \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline \vdots \\ \hline e_j \end{array}\right],$$

where e_j is a row vector in B_{ij}. To argue the correctness of the construction, we write L as a 2×2 block matrix. As each L_{ij} is invertible (otherwise M_{ij} would not be defined), the top-left block entry is invertible. Therefore, we can find two invertible matrices U, V implementing the required row and column operations such that,

$$L = U \left[\begin{array}{cccc|c} L_{11} & 0 & \cdots & 0 & 0 \\ 0 & L_{12} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & L_{mm} & 0 \\ \hline 0 & 0 & \cdots & 0 & \widetilde{D} \end{array}\right] V,$$

for some $m \times m$ matrix $\widetilde{D}$.

Claim 54. *The matrix $\widetilde{D}$ is exactly the input matrix M.*

Proof of Claim. From the 2×2 block decomposition we can write,

$$\widetilde{D} = [C_{11} C_{12} \cdots C_{mm}] \begin{bmatrix} L_{11}^{-1} & 0 & \cdots & 0 \\ 0 & L_{12}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L_{mm}^{-1} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ \vdots \\ B_{mm} \end{bmatrix} = \sum_{i,j} C_{ij} L_{ij}^{-1} B_{ij}.$$

Observe that, for each i, j, $C_{ij} L_{ij}^{-1} B_{ij}$ is an $m \times m$ matrix with M_{ij} as the $(i,j)^{th}$ entry and remaining entries are 0. Hence, $\widetilde{D} = M$. □

Notice that the top-left block of L in Equation 11 is invertible as for each $i, j \in [m]$, L_{ij} is invertible. Now the proof follows from Lemma 52. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il