NON-COMMUTATIVE REPRESENTATIONS OF FAMILIES OF $k^2$
COMMUTATIVE POLYNOMIALS IN $2k^2$ COMMUTING VARIABLES

HARRY DYM, BILL HELTON, AND CALEB MEIER

ABSTRACT. Given a collection $\mathcal{P} = \{p_1(x_1, \ldots, x_{2k^2}), \ldots, p_{k^2}(x_1, \ldots, x_{2k^2})\}$ of $k^2$ commutative polynomials in $2k^2$ variables, the objective is to find a condensed representation for these polynomials in terms of a single non-commutative polynomial $p(X, Y)$ in two $k \times k$ matrix variables $X$ and $Y$. Algorithms that will generically determine whether the given family $\mathcal{P}$ has a non-commutative representation and that will produce such a representation if they exist are developed. These algorithms will determine a non-commutative representation for families $\mathcal{P}$ that admit a non-commutative representation in an open, dense subset of the vector space of non-commutative polynomials in two variables.

CONTENTS

1. Introduction 3
   1.1. Problem statement 3
   1.2. Our Algorithms 4
      1.2.1. Examples 4
   1.3. Effectiveness of our algorithms 6
      1.3.1. General Results 6
      1.3.2. More detailed statements 7
   1.4. Uniqueness 8
      1.4.1. Polynomial identities and non-uniqueness of $p$ 8
      1.4.2. Uniqueness of $X, Y$ 9
   1.5. Motivation 10
   1.6. Computational Cost 11
2. One and two letter monomials in the $k^2$ polynomials: Determining the diagonal variables 11
   2.1. Preliminary calculations 11
   2.2. Enumerating one letter monomials in the $k \times k$ array 14
   2.3. Enumerating one and two letter monomials in the $k \times k$ array 16
      2.3.1. Enumeration of two letter monomials with one letter on the diagonal 19
   2.4. Determining Diagonal Elements 20

Date: December 3, 2012.

Key words and phrases. Non-commutative representations, non-commutative polynomials, generic matrices, rings with polynomial identity.

Helton and Meier were partially funded by NSF Research supported by NSF grants DMS-0700758, DMS-0757212, and the Ford Motor Co. We thank Klep and Vinnikov for valuable conversations.
3. Partitioning Algorithms for families containing single letter monomials $ax^n_i, n \geq 2$

3.1. Preliminary calculations for the partitioning algorithm 22
3.2. Partitioning Algorithm I 24
3.3. Partitioning with homogeneous components: Algorithm DiagPar1 25
3.4. Partitioning: Algorithm DiagPar2 27
3.5. Summary of Partitioning Algorithms 28

4. Positioning algorithms for families of polynomials containing one letter monomials 28

4.1. Positioning the variables within $X$: Algorithm ParPosX 28
4.2. Algorithm for positioning the polynomials given positioning in $X$ 30
4.3. Positioning $Y$ given the position of the polynomials: Algorithm PosY 30
4.4. Uniqueness results for one-letter algorithms 31
4.4.1. Proof of Theorem 1.5 35
4.5. Determining $p(X,Y)$ given $X,Y$ and the positions of the polynomials in $P$ 35

4.5.1. Algorithm NcCoef 35
4.5.2. Homogeneous sorting and Implementation of NcCoef 37
4.5.3. Effectiveness of NcCoef 38
4.6. The Size and Cost of NcCoef 38
4.7. Final Results for One letter Algorithms 39

5. Examples 41

6. Cost of Alg.2 vs. a Brute Force Approach 48
6.1. Brute Force 48
6.2. Cost of Algorithm 2 49
6.2.1. Algorithm DiagA: Determining the Diagonal Entries 49
6.2.2. Cost Estimate for DiagA 50
6.2.3. Cost of DiagPar2 50
6.2.4. Cost of ParPosX 51
6.2.5. Cost of PosPol 51
6.2.6. Cost of PosY 52
6.2.7. Cost of Algorithm 2 52
6.2.8. Comparison of Costs 53

7. Families that may not contain a term of the form $ax^n_i$ with $n > 1$ 53
7.1. Reduction to one letter algorithms: SVR Algorithm 54
7.2. Algorithms based on two letter words 57
7.2.1. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s \geq t + 2$ 57
7.2.2. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s = t + 1$ 58
7.2.3. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s = t$ 58
7.2.4. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v$ 61
7.3. Summary of Two-letter Algorithms 62
7.3.1. Proof of Theorem 1.2 63
7.4. Uniqueness results for two-letter algorithms 64

References 64
1. INTRODUCTION

This paper addresses a new type of problem concerning a condensed description of a collection of polynomials.

1.1. Problem statement. The problem is to represent a family \( P \) of \( k^2 \) polynomials \( p_1, \ldots, p_{k^2} \) in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) as a non-commutative (nc) polynomial \( p = p(X, Y) \) in two \( k \times k \) matrices \( X \) and \( Y \) whose entries are the variables \( x_j \) (without repetition). For example, it is readily checked that if

\[
\begin{align*}
p_1(x_1, \ldots, x_8) &= x_1^2 + x_2x_3 + x_1x_5 + x_2x_7 \\
p_2(x_1, \ldots, x_8) &= x_1x_2 + x_2x_4 + x_1x_6 + x_2x_8 \\
p_3(x_1, \ldots, x_8) &= x_1x_3 + x_3x_4 + x_3x_5 + x_4x_7 \\
p_4(x_1, \ldots, x_8) &= x_2x_3 + x_4^2 + x_3x_6 + x_4x_8, \\
\end{align*}
\]

then

\[
\begin{pmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4
\end{pmatrix} = X^2 + XY \quad \text{with} \quad X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} x_5 & x_6 \\ x_7 & x_8 \end{pmatrix}.
\]

The main objectives of this paper are to:

(1) Present a number of conditions that a given set of polynomials

\[
p_1(x_1, \ldots, x_{2k^2}), \ldots, p_{k^2}(x_1, \ldots, x_{2k^2})
\]

in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) must satisfy in order for it to admit an nc representation \( p(X, Y) \).

(2) Present a number of procedures for recovering such representations, when they exist.

To formally describe the problem we shall say that a family \( P \) of \( k^2 \) polynomials \n
\[
p_1 = p_1(x_1, \ldots, x_{2k^2}), \ldots, p_k = p_k(x_1, \ldots, x_{2k^2}),
\]

in \( 2k^2 \) commutative variables \( x_1, \ldots, x_{2k^2} \), admits a nc representation if there exists a pair of \( k \times k \) matrices \( X \) and \( Y \) and an nc polynomial \( p \) in two nc variables such that

\[
X = \begin{pmatrix}
x_{\sigma(1)} & x_{\sigma(2)} & \cdots & x_{\sigma(k)} \\ 
\vdots & \vdots & \ddots & \vdots \\ 
x_{\sigma(k(k-1)+1)} & x_{\sigma(k(k-1)+2)} & \cdots & x_{\sigma(k^2)} 
\end{pmatrix},
\]

\[
Y = \begin{pmatrix}
x_{\sigma(k^2+1)} & x_{\sigma(k^2+2)} & \cdots & x_{\sigma(k^2+k)} \\ 
\vdots & \vdots & \ddots & \vdots \\ 
x_{\sigma(k^2+k(k-1)+1)} & x_{\sigma(k^2+k(k-1)+2)} & \cdots & x_{\sigma(2k^2)}
\end{pmatrix}
\]
and

\[
p(X, Y) = \begin{pmatrix}
p_{\lambda(1)} & p_{\lambda(2)} & \cdots & p_{\lambda(k)} \\
. & . & . & . \\
. & . & . & . \\
p_{\lambda(k-1)+1} & p_{\lambda(k-1)+2} & \cdots & p_{\lambda(k^2)}
\end{pmatrix},
\]

where \(\sigma\) is a permutation of the set of integers \(\{1, \ldots, 2k^2\}\) and \(\lambda\) is a permutation of the set of integers \(\{1, \ldots, k^2\}\).

1.2. Our Algorithms. The main contribution of this paper is to introduce a collection of algorithms for solving the nc polynomial representation problem. Since they are long, full descriptions are postponed to the body of the paper. However, we shall try to present their flavor in this subsection.

The algorithms are based on the analysis of the patterns of one letter words, two letter words and some three letter words in the given family of polynomials. Thus, for example, if the one letter word \(7x^5\) occurs in one of the polynomials, and the family admits an nc representation \(p(X, Y)\), then \(x^5\) must be a diagonal entry of either \(X\) or \(Y\) and \(k - 1\) of the other polynomials will contain either exactly one or two one letter words of the same degree with the coefficient 7. If there are no one letter words, then the analysis is more delicate; see Sections 7 and 7.2.

If the diagonal variables are determined successfully, then subsequent algorithms serve to partition the remaining \(2k^2 - 2k\) variables between \(X\) and \(Y\) and then to position them within these matrices. En route the \(k^2\) polynomials are arranged in an appropriate order in a \(k \times k\) array. The final step is to obtain the nc polynomial; this is done by matching coefficients as in Example 1.1 below.

Most families of polynomials containing \(k^2\) polynomials in \(2k^2\) variables will not have nc representations. Either there will be no way of partitioning and positioning the variables that is consistent with the family or there will be no choice of coefficients that work.

1.2.1. Examples. The next examples serve to illustrate some of the structure one sees in families of polynomials \(\mathcal{P}\) that admit an nc representation and how it corresponds to diagonal determination, positioning and partitioning.

**Example 1.1.** The family of polynomials

\[
\begin{align*}
p_1 &= 3x_2x_4 + 3x_4x_8 + x_2x_3 + x_4x_7 + 6x_1x_3 + 6x_3x_7 + x_1x_4 + x_3x_8 \\
p_2 &= 3x_2x_6 + 3x_6x_8 + x_1x_6 + x_5x_8 + 6x_1x_5 + 6x_5x_7 + x_2x_5 + x_6x_7 \\
p_3 &= 3x_7^2 + 2x_1x_2 + 3x_4x_6 + x_4x_5 + x_3x_6 + 6x_1^2 + 6x_3x_5 \\
p_4 &= 3x_4x_6 + 3x_8^2 + 2x_7x_8 + x_3x_6 + 6x_3x_5 + 6x_7^2 + x_4x_5
\end{align*}
\]

admits an nc representation.
**Discussion:** If the given family of polynomials admits an nc representation \( p(X, Y) \), then it must admit at least one representation of the form

\[
p(X, Y) = aX^2 + bXY + cYX + dY^2
\]

for some choice of \( a, b, c, d \in \mathbb{R} \), since \( p_1, \ldots, p_4 \) are homogeneous of degree two. Moreover, as

\[
p_3 = 3x_2^2 + 6x_1^2 + 2x_1x_2 + \cdots \quad \text{and} \quad p_4 = 3x_8^2 + 6x_7^2 + 2x_7x_8 + \cdots
\]

and there are no other one letter words in the family \( P \), it is not hard to see (as we shall clarify later in more detail in §2) that \( x_2^2 \) and \( x_8^2 \) are diagonal entries in one of the matrices, say \( X \), and correspondingly \( x_1^2 \) and \( x_7^2 \) are diagonal entries in the other matrix, \( Y \). Moreover, since \( x_2^2 \) and \( x_6^2 \) are in \( p_3 \), whereas \( x_7^2 \) and \( x_8^2 \) are in \( p_4 \), it follows that if an nc representation exists, then, either

\[
X = \begin{pmatrix} x_2 & ? \\ ? & x_8 \end{pmatrix}, \quad Y = \begin{pmatrix} x_1 & ? \\ ? & x_7 \end{pmatrix} \quad \text{and} \quad p(X, Y) = \begin{pmatrix} p_3 & ? \\ ? & p_4 \end{pmatrix},
\]

or

\[
X = \begin{pmatrix} x_8 & ? \\ ? & x_2 \end{pmatrix}, \quad Y = \begin{pmatrix} x_7 & ? \\ ? & x_1 \end{pmatrix} \quad \text{and} \quad p(X, Y) = \begin{pmatrix} ? & ? \\ p_4 & ? \end{pmatrix},
\]

and, in (1.4) we must have

\[
a = 3, \quad (b + c) = 2 \quad \text{and} \quad d = 6.
\]

We shall assume that (1.5) holds; the other possibility may be treated similarly.

The next step is to try to **partition** the remaining variables between \( X \) and \( Y \). Towards this end it is useful to note that if

\[
X = \begin{pmatrix} x_2 & x_a \\ x_b & x_8 \end{pmatrix}
\]

then \( X^2 = \begin{pmatrix} x_2^2 + x_4x_6 & x_2x_8 + x_6x_8 \\ x_4x_2 + x_8x_6 & x_4^2 + x_6x_8 \end{pmatrix} \),

and hence (since we are assuming that (1.5) is in force) that \( p_3 \) must contain a term of the form \( 3x_4x_6 \). Comparison with the given polynomial \( p_3 \) leads to the conclusion that \( x_4 \) and \( x_6 \) belong to \( X \). Let us arbitrarily **position** \( x_6 \) as the 12 entry of \( X \) and \( x_4 \) as the 21 entry of \( X \). Then

\[
X = \begin{pmatrix} x_2 & x_6 \\ x_4 & x_8 \end{pmatrix}, \quad \text{and} \quad X^2 = \begin{pmatrix} x_2^2 + x_6x_4 & x_2x_6 + x_6x_8 \\ x_4x_2 + x_8x_6 & x_4^2 + x_6x_8 \end{pmatrix}.
\]

The 11 entry of \( X^2 \) provides no new information, but comparison of the 12 entry with the given polynomials leads to the conclusion that if the given family admits an nc representation, then \( p_2 \) must sit in the 12 position in \( p(X, Y) \) and hence

\[
p(X, Y) = \begin{pmatrix} p_3 & p_2 \\ p_1 & p_4 \end{pmatrix}.
\]

Similarly,

\[
Y = \begin{pmatrix} x_1 & x_c \\ x_d & x_7 \end{pmatrix} \implies Y^2 = \begin{pmatrix} \cdot & x_1x_c + x_cx_7 \\ \cdot & \cdot \end{pmatrix},
\]
which upon comparison with the entries in $p_2$ leads to the conclusion that $x_c = x_5$. Therefore, $x_d = x_3$. Comparison of

$$3X^2 + bXY + cYX + 6Y^2$$

with

$$p = \begin{bmatrix} p_3 & p_2 \\ p_1 & p_4 \end{bmatrix}$$

implies further that equality will prevail if and only $b = c = 1$ (i.e., $a = 3$, $b = c = 1$ and $d = 6$ in (1.4)).

The example illustrates the strategy of first determining which variables occur on either the diagonal of $X$ or of $Y$. In general, if the given family $P$ admits an nc representation $p(X,Y)$ of degree $d$ and if

$$p(X,Y) = aX^n + bY^n + \cdots$$

with $|a| + |b| > 0$

(and no other nonzero multiples of $X^n$ and $Y^n$) for some positive integer $n \geq 2$, then:

1. if $b = 0$ (resp., $a = 0$), there will be exactly $k$ one letter words of degree $n$ with coefficient $a$ (resp., $b$);

2. if $ab \neq 0$ and $a \neq b$, there will be exactly $k$ one letter words of degree $n$ with coefficient $a$ and exactly $k$ one letter words of degree $n$ with coefficient $b$;

3. if $a = b$, there will be exactly $2k$ one letter words of degree $n$ with coefficient $a$.

Example 1.1 fits into setting (2).

So far we have focused on how we can use one letter words occurring in polynomials in $P$. A substantial part of this paper is also devoted to developing procedures for finding the diagonal variables that are based on patterns in two and (some) three letter words. The latter come into play if there are no one letter words to partition the diagonal variables between $X$ and $Y$.

1.3. Effectiveness of our algorithms. Our algorithms depend upon the existence of appropriate patterns of one, two and some three letter words in the $k^2$ polynomials in the given family. We shall show that these algorithms are effective generically, i.e., they are effective on an open dense set of the set of polynomials that admit nc representations.

1.3.1. General Results. Let $W$ be the space of nc polynomials in two variables of degree $d$. We say that a subspace $U$ of $W$ is of degree $d$ if the maximum degree of the basis elements of $U$ is $d$.

**Theorem 1.2.** Let $p_1, \ldots, p_{k^2}$ be a family $P$ of polynomials in $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ of degree $d > 3$ and let $U$ be a subspace of $W$ of degree $d$. Then there exists an open dense subset $S$ of $U$ for which the algorithms developed in this paper determine an nc representation $p \in S$ for $P$ if and only if $P$ has an nc representation $p \in S$. If such a representation exists, then these algorithms may be used to construct it.

**Proof.** The proof is postponed until Section 7.3. □
Example 1.3, below, exhibits a family \( \mathcal{P} \) with an nc representation \( p \) for which the algorithms do not work. However, perturbing \( p \) gives \( \hat{p} \) for which they do.

**Example 1.3.** Suppose that we are given the list of polynomials

\[
\begin{align*}
p_1 &= x_2x_4 + x_4x_8 + x_2x_3 + x_4x_7 + x_1x_3 + x_3x_7 + x_1x_4 + x_3x_8 \\
p_2 &= x_2x_6 + x_6x_8 + x_1x_6 + x_5x_8 + x_1x_5 + x_5x_7 + x_2x_5 + x_6x_7 \\
p_3 &= x_2^2 + 2x_1x_2 + x_4x_6 + x_4x_5 + x_3x_6 + x_1^2 + x_3x_5 \\
p_4 &= x_4x_6 + x_3^2 + 2x_7x_8 + x_3x_6 + x_3x_5 + x_7^2 + x_4x_5
\end{align*}
\]

**Discussion** If the given family admits an nc representation \( p(X, Y) \), then it is readily seen from the one letter words in the family \( \mathcal{P} \) that it must be of the form (1.4) with \( a = d = 1 \) and that \( x_1, x_2, x_7 \) and \( x_8 \) are diagonal variables.

However, it is impossible to decide on the basis of one letter words which of these variables belong to \( X \) and which belong to \( Y \). The most that we can say so far is that \( x_1, x_2 \) and \( p_3 \) must lie in the same diagonal position, and hence \( x_7, x_8 \) and \( p_4 \) are in the other diagonal position. Thus, we may assume that \( x_2 \) is in the 11 position of \( X \), \( x_1 \) is in the 11 position of \( Y \). But it is still not clear how to allocate \( x_7 \) and \( x_8 \).

It is readily seen that \( p(X, Y) = X^2 + XY + YX + Y^2 \) is an nc representation for the family of polynomials in Example 1.3. Thus, \( p(X, Y) \) is contained in the subspace \( \mathcal{U} \) of nc polynomials defined by

\[
\mathcal{U} = \{aX^2 + bXY + cYX + dY^2 : a, b, c, d \in \mathbb{R}\},
\]

and, although our algorithms are not effective on the entire subspace \( \mathcal{U} \), they do work on the (open dense) subset \( \mathcal{S} \) of \( \mathcal{U} \) consisting of polynomials in \( \mathcal{U} \) for which \( a \neq d \) and \( 2a \neq b \).

1.3.2. More detailed statements. Our main theorems on algorithm effectiveness are more detailed than Theorem 1.2. These theorems and a number of our algorithms depend in part on the coefficients of the terms in the **commutative collapse** \( \hat{p} \) of an nc polynomial \( p(X, Y) \), which is defined as the commutative polynomial

\[
\hat{p}(x, y) = p(xI, yI).
\]

In particular, if \( \varphi(i, j) \) is the sum of the coefficients of the terms in the nc polynomial \( p(X, Y) \) of degree \( i \) in \( X \) and degree \( j \) in \( Y \), then \( \varphi(i, j) \) is the coefficient of \( x^iy^j \) in \( \hat{p}(x, y) \).

We shall also need the following more refined quantities:

\( \varphi(i, j; X) \) (resp. \( \varphi(i, j; Y) \)) denotes the sum of the coefficients of the terms in the nc polynomial \( p(X, Y) \) of degree \( i \) in \( X \) and degree \( j \) in \( Y \) that end in \( X \) (resp. end in \( Y \));

\( \varphi(X; i, j) \) (resp. \( \varphi(Y; i, j) \)) denotes the sum of the coefficients of the terms in the nc polynomial \( p(X, Y) \) of degree \( i \) in \( X \) and degree \( j \) in \( Y \) that begin with \( X \) (resp. begin with \( Y \)).

Thus, for example, if

\[
p(X, Y) = aX^2YXY + bXYXYX + cYXYX^2 + dYX^3Y,
\]
then
\[ \varphi(X; 3, 2) = a + b, \quad \varphi(Y; 3, 2) = c + d, \quad \varphi(3, 2; X) = b + c \quad \text{and} \quad \varphi(3, 2; Y) = a + d. \]

Clearly
\[ \varphi(i, j) = \varphi(X; i, j) + \varphi(Y; i, j) = \varphi(i, j; X) + \varphi(i, j; Y). \]

The next theorem provides some insight into our one letter algorithms. There is an analogous result for two letter words: Theorem 7.14, which will be presented in Section 7.3

**Theorem 1.4.** Let \( p_1, \ldots, p_{2k} \) be a family \( \mathcal{P} \) of polynomials in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k} \) and let \( \mathcal{W} \) denote the set of nc polynomials \( p(X, Y) \) of degree \( d > 1 \) such that there exists an integer \( t \geq 2 \) for which
\[ \varphi(t, 0) \neq 0, \quad \varphi(0, t) \neq 0, \quad \varphi(t, 0) \neq \varphi(0, t) \]
and that additionally satisfies one of the following properties:
(1) \( t\varphi(t, 0) \neq \varphi(t - 1, 1) \)
(2) \( t\varphi(0, t) \neq \varphi(1, t - 1) \)
(3) \( \varphi(t - 1, 1; Y) \neq 0 \) and \( \varphi(t, 0) \neq \varphi(t - 1, 1; Y) \)
(4) \( \varphi(Y; t - 1, 1) \neq 0 \) and \( \varphi(t, 0) \neq \varphi(Y; t - 1, 1) \)
(5) \( \varphi(1, t - 1; X) \neq 0 \) and \( \varphi(0, t) \neq \varphi(1, t - 1; X) \)
(6) \( \varphi(X; 1, t - 1) \neq 0 \) and \( \varphi(0, t) \neq \varphi(X; 1, t - 1) \).

Then the one letter algorithms stated in Section 4.7 determine that \( \mathcal{P} \) admits an nc representation \( p(X, Y) \) in the class \( \mathcal{W} \) if and only if \( \mathcal{P} \) has a representation in this class. If such a representation exists, then these algorithms can be used to construct it.

**Proof.** See Theorems 4.14, 4.15 and Remark 4.16. \( \square \)

The long list of caveats looks confining, but they are all strict inequality constraints and so hold generically.

1.4. **Uniqueness.** The issue of uniqueness of an nc representation is of interest in its own right. We shall see is that while the representation \( p(X, Y) \) is highly non-unique, the arrangement of commutative variables \( x_j \) in the matrices \( X \) and \( Y \) is determined up to permutations, transpositions and interchanges of \( X \) and \( Y \).

1.4.1. **Polynomial identities and non-uniqueness of \( p \).** A basic theorem in the theory of rings with polynomial identities implies that if \( \Sigma_{2k} \) denotes the set of all permutations of the set \( \{1, \ldots, 2k\} \) for each positive integer \( k \), then the polynomial
\[ q(X_1, \ldots, X_{2k}) = \sum_{\sigma \in \Sigma_{2k}} \text{sgn}(\sigma) X_{\sigma(1)} \cdots X_{\sigma(2k)} = 0 \]
for every choice of the \( k \times k \) matrices \( X_1, \ldots, X_{2k} \) in \( \mathbb{R}^{k \times k} \). Thus, if \( X \) and \( Y \) are arbitrary real \( k \times k \) matrices and if
\[ p(X, Y) \overset{\text{def}}{=} q(X_1, \ldots, X_{2k}) \]
with
\[ X_j = [X^j, Y] \quad \text{for} \quad j = 1, \ldots, 2k, \]
then \( p(X, Y) = 0 \); see [3] and [1] for additional information. Any other replacement of \( X_j \) in (1.7) by a polynomial in \( X \) and \( Y \) will also yield a polynomial \( p(X, Y) = 0 \). However, the choice in (1.9) will have nonzero coefficients. In particular this means that if a given family \( \mathcal{P} \) of polynomials admits an nc representation, then it admits infinitely many nc representations.

If \( k = 2 \), for example, the nc polynomials

\[
(1.10) \quad YXY^2X + Y^2X^2Y + YX^2YX + XYY^2X + XYXY^2 + X^2YXY
\]

and

\[
(1.11) \quad Y^2XYX + YX^2Y^2 + XY^2XY + YXYX^2 + XYXY^2 + X^2Y^2X
\]

generate the same family regardless of how the commutative variables \( x_1, \ldots, x_8 \) are partitioned between \( X \) and \( Y \) and positioned.

This stems from the fact that the difference between the nc polynomial in (1.10) and the nc polynomial in (1.11) is equal to the commutator

\[
[Y - X, (XY - YX)^2] = (Y - X)(XY - YX)^2 - (XY - YX)^2(Y - X) = 0,
\]

since for \( 2 \times 2 \) matrices

\[
X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} x_5 & x_6 \\ x_7 & x_8 \end{pmatrix},
\]

the polynomial \((XY - YX)^2\) has the special form

\[
(1.12) \quad p(X, Y) = [X, Y]^2 = (XY - YX)^2 = \begin{pmatrix} p(x) & 0 \\ 0 & p(x) \end{pmatrix}.
\]

This is well known by the experts in matrix identities, and it is easily verified by direct calculation that

\[
p(x) = x_2^2x_7^2 - x_2\{x_3[2x_6x_7 + (x_5 - x_8)^2] - (x_1 - x_4)x_7(x_5 - x_8)\}
+ x_6\{x_1x_3x_5 - x_3x_4x_5 + x_2^2x_6 - x_4^2x_7 + 2x_1x_4x_7 - x_4^2x_7 + x_3(-x_1 + x_4)x_8\},
\]

Thus, the polynomial \( p(X, Y) \) in (1.12) is an example of a homogeneous nc polynomial that produces a family \( \mathcal{P} \) with some of the polynomials in \( \mathcal{P} \) equal to zero and some not. We shall see later that the fact that the degrees of the polynomials in \( \mathcal{P} \) are either 4 or 0 is consistent with Lemma 4.7.

If \( p(X, Y) = (X + Y)^n \) for some positive integer \( n \), then it is impossible to determine which variables belong to \( X \) and which variables belong to \( Y \).

The theorems presented later in the paper that validate our algorithms, e.g., Theorem 1.4, have hypotheses that exclude cases like (1.10).

1.4.2. Uniqueness of \( X, Y \). We just saw that an nc representation for \( \mathcal{P} \) is highly non-unique, however, the pair \( X, Y \) in such representations is generically very tightly determined. This is indicated by the following theorem.

**Theorem 1.5.** If \( \mathcal{P} \) is a family of polynomials \( p_1, \ldots, p_k \) in the commutative variables \( x_1, \ldots, x_{2k^2} \) that admits two nc representations \( p(X, Y) \) and \( \tilde{p}(\tilde{X}, \tilde{Y}) \) that satisfy the conditions of Theorem 1.4, then there exists a permutation matrix \( \Pi \) such that one of
the following must hold:

\begin{align*}
(1) & \quad X = \Pi^T \tilde{X} \Pi, \quad Y = \Pi^T \tilde{Y} \Pi, \\
(2) & \quad X = \Pi^T \tilde{X}^T \Pi, \quad Y = \Pi^T \tilde{Y}^T \Pi, \\
(3) & \quad X = \Pi^T \tilde{Y} \Pi, \quad Y = \Pi^T \tilde{X} \Pi, \\
(4) & \quad X = \Pi^T \tilde{Y}^T \Pi, \quad Y = \Pi^T \tilde{X}^T \Pi.
\end{align*}

(1.13)

Proof. The proof is postponed until Section 4.4.

We shall say that the pairs \(X, Y\) and \(\tilde{X}, \tilde{Y}\) are permutation equivalent if they are related by any of the four choices in (1.13).

1.5. Motivation. The problem we study in the paper is undertaken primarily for its own sake, however, the original motivation arose from the observation that the running time for algebraic calculations on a large family of commutative polynomials \(P\) can be much longer than the corresponding calculation on a small family of nc polynomials representing \(P\). Such calculations can be done using nc computer algebra, for example NCAlgebra or NCGB [2], which runs under Mathematica.

As an example, consider computing Gröbner Bases, a powerful but time consuming algebraic construction. The reader does not need to know anything about them to get the thrust of this example. We have a list \(P\) of nc polynomials and run an nc Gröbner Basis algorithm on

\(P = \{a^T m + m^T a + m^T m, aw + w^T w + w^T a^T, m^T am,
\quad m^T aw, m^T a^T m, w^T aw, w^T a^T m, w^T a^T w\}\).

Using NCGB on a Macbook Pro it finished in .007 seconds. Now substitute two by two matrices

\[
a \to \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad w \to \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}, \quad m \to \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}
\]

with commuting entries for the variables and run the ordinary Mathematica Gröbner Basis Command. The run took 161 seconds in the most favorable monomial order that we tried. The corresponding \(3 \times 3\) matrix substitution yielded a Gröbner Basis computation which did not finish in 1 hour. Calculation with higher order matrix substitutions would be prohibitive.

The NC Gröbner Basis and the commutative one contain different information. Namely, the NCGB determines membership in the two sided ideal \(I_P\) generated by \(P\) while the commutative GB obtained from “generic” \(n \times n\) matrix substitution determines membership in the ideal generated by \(I_P + R_n\) where \(R_n\) is the ideal of all nc polynomials which vanish on the \(n \times n\) matrices. We would assert that the NCGB contains very valuable information (possibly more than in the \(I_P + R_n\) case) and is readily obtained. In fact what brought us to the nc polynomial representation question was the reverse side of this. To speed up nc GB runs we tried symbolic matrix substitutions in the hope that the commutative GBs would go quickly and as \(n\) got bigger guide us toward an NCGB. This approach seems hopeless because of prohibitively long run times.
In special circumstances nc representations could have a significant advantage for numerical computation. In particular, the numerics for solving the second order polynomial (in matrices) equation, called a Riccati equation, is highly developed. Consequently, it would be very useful to be able to replace a collection of conventional polynomial equations by an nc representation.

Finally we mention that there is a burgeoning area devoted to extending real (and some complex) algebraic geometry to free algebras. Here one analyzes non-commutative polynomials with properties determined by substituting in square matrices of arbitrary size. See the recent references [4, 5, 6, 8, 9] and their extensive bibliographies.

1.6. **Computational Cost.** The problem considered here can be attacked by “brute force” rather than by the methods developed in this paper. There are \((2k^2)!\) arrangements of the variables \(x_1, \ldots, x_{2k^2}\) in \(X\) and \(Y\) and \((k^2)!\) arrangements of the polynomials \(p_1, \ldots, p_{k^2}\) in \(P\). For a given arrangement \(\sigma\) of the variables in \(X\) and \(Y\), one obtains a matrix of commutative polynomials by forming a general nc polynomial \(p(X,Y) = \sum_{\alpha,\beta} c_{\alpha\beta} m_{\alpha\beta}(X,Y)\) of degree \(d\) as in (2.3) with undetermined coefficients \(c_{\alpha\beta}\) that are chosen to match the array determined by the arrangement \(\lambda\) of \(k^2\) polynomials, if possible. For each pair of arrangements \(\sigma\) and \(\lambda\), one attempts to solve for the coefficients \(c_{\alpha\beta}\) to obtain an nc representation. We will refer to this approach as the **Brute Force Method**. Because there are \((k^2)!(2k^2)!\) possible systems, the cost of this approach is very high. Also, to rule out the existence of an nc representation this way it is necessary to **check all of these cases and to verify that they fail**.

Much to the contrary, the procedures we introduce are likely to determine non-existence in the first few step. Even when me must run through all the steps, we find that the implementation of the algorithm that we call Algorithm 2 requires on the order of

\[
10 \left( k^7 + 3dk^5 + d^3k^3 \right) + \sum_{i=2}^{d} \frac{2^{3i+1}}{3}
\]

operations, which is much less than the

\[
(2k^2)!(k^2)! \left( \sum_{i=2}^{d} \frac{2^{3i+1}}{3} \right)
\]

operations required by brute force; see §6 for details.

2. **One and Two Letter Monomials in the \(k^2\) Polynomials: Determining the Diagonal Variables**

In this section we enumerate the one and two letter monomials that appear in the \(k \times k\) array of commutative polynomials corresponding to the nc polynomial \(p(X,Y)\).

2.1. **Preliminary calculations.** This subsection is devoted to notation and a couple of definitions that will be useful in the main developments.

Let \(e_1, \ldots, e_k\) denote the standard basis for \(\mathbb{R}^k\) and let \(E_{st}\) denote the \(k \times k\) matrix with a 1 in the \(st\) position and 0’s elsewhere. Then, since

\[
E_{st} = e_s e_t^T,
\]
it is readily seen that
\[ E_{st}E_{uv} = e_s(e^T_t e_u)e^T_v = \begin{cases} 0 & \text{if } t \neq u \\ E_{uv} & \text{if } t = u \end{cases} \]
and hence that
\[ (E_{st})^2 = \begin{cases} 0 & \text{if } s \neq t \\ E_{st} & \text{if } s = t \end{cases} \]

Let \( \alpha = (\alpha_1, \ldots, \alpha_\ell), \beta = (\beta_1, \ldots, \beta_\ell) \) be multi-indices with positive integer entries, except for \( \beta_\ell \), which may also be zero and suppose further that
\[ \alpha_1 + \cdots + \alpha_\ell = s, \quad \beta_1 + \cdots + \beta_\ell = t, \]
and let
\[ m_{\alpha,\beta}(X, Y) = X^{\alpha_1}Y^{\beta_1} \cdots X^{\alpha_\ell}Y^{\beta_\ell}. \]
Then, since \( s \geq \ell \) and \( t \geq \ell - 1 + \beta_\ell \), it follows that
\[ \ell \leq s \quad \text{and} \quad \ell \leq t + 1 - \beta_\ell. \]
The proof of the next two lemmas will rest heavily on the following observations:

if \( m, r \) and \( n \) are nonnegative integers such that \( m + r \geq 2 \) and \( n \geq 2 \), then
\[ (E_{cd})^m(E_{ab})^n(E_{cd})^r = \begin{cases} E_{aa} & \text{if } a = b = c = d \\ 0 & \text{otherwise} \end{cases}. \]
(2.4)

\[ E_{aa}E_{cd}E_{aa} = \begin{cases} E_{aa} & \text{if } c = d = a \\ 0 & \text{otherwise} \end{cases}. \]
(2.5)

\[ E_{ab}E_{cc}E_{ab} = \begin{cases} E_{aa} & \text{if } c = d = a \\ 0 & \text{otherwise} \end{cases}. \]
(2.6)

**Remark 2.1.** It is also useful to note that if the constraints (2.1) and (2.2) are in force and
\[ X = x_iE_{ab} + \cdots \quad \text{and} \quad Y = x_jE_{cd} + \cdots, \]
then
\[ m_{\alpha,\beta}(X, Y) = x_i^s x_j^t m_{\alpha,\beta}(E_{ab}, E_{cd}) + \cdots. \]
(2.8)

**Lemma 2.2.** Assume that the multi-indices \( \alpha = (\alpha_1, \ldots, \alpha_\ell) \) and \( \beta = (\beta_1, \ldots, \beta_\ell) \) are subject to the constraints (2.1) and (2.2). Suppose further that \( s \geq 2, t \geq 2, \) and that
\[ \max\{\alpha_1, \ldots, \alpha_\ell; \beta_1, \ldots, \beta_\ell\} \geq 2. \]
(2.9)
Then
\[ m_{\alpha,\beta}(x_iE_{ab}, x_jE_{cd}) = \begin{cases} x_i^s x_j^t E_{aa} & \text{if } a = b = c = d \\ 0 & \text{otherwise} \end{cases}. \]
(2.10)
In other words, \( m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) \neq 0 \) if and only if \( x_i \) and \( x_j \) are diagonal pairs in the same position.

Proof. The proof is divided into cases.

1. If \( \ell = 1 \), then
   \[
   m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) = x_i^s x_j^t (E_{ab})^s (E_{cd})^t
   \]
   and the asserted conclusion (2.10) follows from (2.4) with \( m = s \) and \( n = t \), since \( s \geq 2 \) and \( t \geq 2 \), by assumption.

2. If \( \ell > 1 \) and \( \alpha_r \geq 2 \) for some \( r \in \{1, \ldots, k\} \), then
   \[
   (x_i E_{ab} + \cdots)^r = x_i^r (E_{ab})^r + \cdots = \begin{cases} x_i^r E_{aa} + \cdots & \text{if } b = a \\ 0 & \text{otherwise} \end{cases}
   \]
   But if \( b = a \), then
   \[
   m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) = x_i^s x_j^t (E_{aa})^{\alpha_1} (E_{cd})^{\beta_1} (E_{aa})^{\alpha_2} \cdots
   \]
   and (2.10) follows from (2.5).

3. If \( \ell > 1 \) and \( \beta_r \geq 2 \) for some \( r \in \{1, \ldots, k\} \), then
   \[
   (x_j E_{cd})^r = x_j^r (E_{cd})^r = \begin{cases} x_j^r E_{cc} & \text{if } d = c \\ 0 & \text{otherwise} \end{cases}
   \]
   But if \( d = c \), then
   \[
   m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) = X^{\alpha_1} Y^{\beta_1} X^{\alpha_2} \cdots = x_i^s x_j^t (E_{ab})^{\alpha_1} (E_{cc}) (E_{ab})^{\alpha_2} \cdots
   \]
   and (2.10) follows from (2.6).

Remark 2.3. Condition (2.9) is automatically met if either
   \[
   s > \ell, \quad \text{or} \quad t > \ell, \quad \text{or} \quad s = t = \ell \text{ and } \beta_\ell = 0.
   \]
   It remains to consider the case
   \[
   \max\{\alpha_1, \ldots, \alpha_\ell; \beta_1, \ldots, \beta_\ell\} \leq 1.
   \]

Lemma 2.4. If (2.1), (2.2) and (2.11) are in force and \( t \geq 2 \), then there are four possibilities:

1. \( \beta_\ell = 1 \): In this setting \( s = t, \ell = s \), \( m_{\alpha,\beta}(X, Y) = (XY)^t \) and
   \[
   m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) = x_i^s x_j^t (E_{ab} E_{cd})^t = \begin{cases} x_i^s x_j^t E_{aa} & \text{if } c = b \text{ and } d = a \\ 0 & \text{otherwise} \end{cases}
   \]

2. \( \beta_\ell = 0 \): In this setting \( s = t + 1, \ell = s \), \( m_{\alpha,\beta}(X, Y) = (XY)^t X \) and
   \[
   m_{\alpha,\beta}(x_i E_{ab}, x_j E_{cd}) = x_i^s x_j^t (E_{ab} E_{cd})^t E_{ab} = \begin{cases} x_i^s x_j^t E_{ab} & \text{if } c = b \text{ and } d = a \\ 0 & \text{otherwise} \end{cases}
   \]
3. $\beta_\ell = 1$: In this setting $s = t$, $\ell = s$, $m_{\alpha,\beta}(Y, X) = (YX)^t$ and

$$m_{\alpha,\beta}(x_j E_{cd}, x_i E_{ab}) = x_i^s x_j^t (E_{cd} E_{ab})^t = \begin{cases} x_i^s x_j^t E_{ba} & \text{if } c = b \text{ and } d = a \\ 0 & \text{otherwise} \end{cases}.$$

4. $\beta_\ell = 0$: In this setting $s = t + 1$, $\ell = s$, $m_{\alpha,\beta}(Y, X) = (YX)^t Y$ and

$$m_{\alpha,\beta}(x_j E_{cd}, x_i E_{ab}) = x_i^s x_j^t (E_{cd} E_{ab})^t = \begin{cases} x_i^s x_j^t E_{bb} & \text{if } c = b \text{ and } d = a \\ 0 & \text{otherwise} \end{cases}.$$

**Proof.** In view of (2.1), the constraint (2.11) implies that

$$\alpha_1 = \cdots = \alpha_\ell = 1, \quad \beta_1 = \cdots = \beta_{\ell-1} = 1 \quad \text{and } \beta_\ell = 1 \text{ or } \beta_\ell = 0.$$

Correspondingly

$$m_{\alpha,\beta}(X, Y) = \begin{cases} (XY)^\ell & \text{if } \beta_\ell = 1 \\ (XY)^{\ell-1} X & \text{if } \beta_\ell = 0 \end{cases}$$

and

$$m_{\alpha,\beta}(Y, X) = \begin{cases} (YX)^\ell & \text{if } \beta_\ell = 1 \\ (YX)^{\ell-1} Y & \text{if } \beta_\ell = 0 \end{cases}.$$

The remaining conclusions are self-evident. □

**Definition 2.5.** The two $r$ letter monomials $e x_{i_1}^{\alpha_1} \cdots x_{i_r}^{\alpha_r}$ and $f x_{j_1}^{\beta_1} \cdots x_{j_r}^{\beta_r}$ with $e \neq 0$ and $f \neq 0$ are said to be $\triangleright$-equivalent if there exists a permutation $\sigma$ of the integers $\{1, \ldots, r\}$ such that $\beta_j = \alpha_{\sigma(j)}$ for $j = 1, \ldots, r$. This will be indicated by writing

$$e x_{i_1}^{\alpha_1} \cdots x_{i_r}^{\alpha_r} \triangleright f x_{j_1}^{\beta_1} \cdots x_{j_r}^{\beta_r}.$$

These two monomials are structurally equivalent (SE) if they are $\triangleright$-equivalent and $e = f$. This will be indicated by writing

$$e x_{i_1}^{\alpha_1} \cdots x_{i_r}^{\alpha_r} \quad SE \quad f x_{j_1}^{\beta_1} \cdots x_{j_r}^{\beta_r}.$$

Thus, for example, if $a, b, c, d \in \mathbb{R} \setminus \{0\}$, then the four two letter words

$$a x_1 x_3^4, \quad b x_2^3 x_4^4, \quad c x_2^3 x_4^4 \quad \text{and} \quad d x_5^2 x_6^4$$

are $\triangleright$-equivalent; they will be SE if and only if $a = b = c = d$.

2.2. **Enumerating one letter monomials in the $k \times k$ array.**

**Lemma 2.6.** If a family of polynomials $p_1, \ldots, p_{k^2}$ in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ admits an nc representation $p(X, Y)$, then for each positive integer $n > 1$ exactly one of the following situations prevails:

1. There are no one letter monomials of degree $n$ in any one of the given polynomials.
2. At least one of the given polynomials contains exactly one one letter monomial of degree $n$.
3. At least one of the given polynomials contains exactly two one letter monomials of degree $n$. 


Moreover,

(2) holds $\iff$ there exist exactly $k$ polynomials each one of which contains exactly one one letter monomial $ex^n_{is}$ of degree $n$ (all with the same coefficient).

(3) holds $\iff$ there exist exactly $k$ polynomials each one of which contains exactly two one letter monomials $ex^n_{is} + fx^n_{jt}$.

Further, if $e \neq f$ and $ef \neq 0$ then the letters $x_{in}$, $m = 1, \ldots, k$ in the monomials $ex^n_{i1}, \ldots, ex^n_{ik}$ are the diagonal entries of one of the matrices and the letters $x_{jn}$, $m = 1, \ldots, k$ in the monomials $fx^n_{j1}, \ldots, fx^n_{jk}$ are the diagonal entries of the other.

Proof. Clearly (1), (2) and (3) are mutually exclusive possibilities that correspond to

$$p(X, Y) = aX^n + bY^n + \cdots$$

with either (1) $a = 0$ and $b = 0$ for all $n$, (2) $ab = 0$ and $a \neq b$.

Suppose first that at least one of the given $k^2$ polynomials contains exactly one term of the form $ax^n_{is}$ with $n > 1$, a real coefficient $a \neq 0$ and $x_i \in X$. Then

$$X = x_iE_{ss} + \cdots \text{ for some } s \in \{1, \ldots, k\}$$

and

$$aX^n = a(x^n_{is}E_{ss} + \cdots).$$

Moreover, since the polynomial that contains the term $ax^n_{is}$ contains only one term of this form, it follows that

$$p(X, Y) - aX^n \text{ does not contain a term of the form } cY^n \text{ with } c \neq 0.$$  

Thus, as $X$ has $k$ diagonal entries, there will be exactly $k$ polynomials each one of which contains a exactly one term of this form. This completes the proof of (a). The proof of (b) is similar to the proof of (a).

Finally, a term of the form $ax^n_{is} + bx^n_{jt}$ with $i \neq j$ and $ab \neq 0$ will be present in one of the polynomials if and only if either

$$X = x_iE_{ss} + \cdots \text{ and } Y = x_jE_{ss} + \cdots$$

for some choice of $s \in \{1, \ldots, k\}$, or

$$X = x_jE_{ss} + \cdots \text{ and } Y = x_iE_{ss} + \cdots$$

for some choice of $s \in \{1, \ldots, k\}$. The rest of the proof goes through much as before.

\[\square\]

Lemma 2.7. Let $p_1, \cdots, p_{k^2}$ be a family of polynomials with an nc representation $p(X, Y)$. Suppose that some polynomial $p$ in the family contains the terms $ax^n_{is} + bx^n_{jt}$ with $a \neq b$ and either $a \neq 0$ or $b \neq 0$. Then there exist $k$ polynomials $p_{i1}, \cdots, p_{ik}$ containing the respective terms $ax^n_{i1} + bx^n_{j1}, \cdots, ax^n_{ik} + bx^n_{jk}$. Moreover, the terms $ax^n_{im}$ with $(1 \leq m \leq k)$ that have coefficient $a$ are the diagonal variables of one matrix and the terms with $bx^n_{jm}$ with $(1 \leq m \leq k)$ that have coefficient $b$ are the diagonal terms of the other matrix.

Proof. This follows from Lemma 2.6.  \[\square\]
2.3. **Enumerating one and two letter monomials in the $k \times k$ array.** The symbol

$$\chi(a) = \begin{cases} 
1 & \text{if } a \neq 0 \\
0 & \text{if } a = 0 
\end{cases}$$

will be used in the next lemma.

**Lemma 2.8.** Let $p_1, \ldots, p_{2k^2}$ be a family of polynomials in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ that admits an nc representation $p(X,Y)$ of degree $d > 1$. Suppose further that $s = t, t \geq 2$,

$$p(X,Y) = e_1(XY)^t + e_2(YX)^t + e_3X^{2t} + e_4Y^{2t} + q(X,Y)$$

(2.12)

where $q$ is a polynomial that does not contain any scalar multiples of the first four monomials listed in (2.12), that the coefficients $e_1, \ldots, e_4$ are all distinct and that $x_u \neq x_v$. Then the family of $k^2$ polynomials will contain

- $k^2 - k$ 2 letter monomials SE to $e_1x_u^tx_v^t$ if $e_1 \neq 0$
- $k^2 - k$ 2 letter monomials SE to $e_2x_u^tx_v^t$ if $e_2 \neq 0$
- $k^2 - k$ 2 letter monomials SE to $e_3x_u^tx_v^t$ if $e_3 \neq 0$
- $k^2 - k$ 2 letter monomials SE to $e_4x_u^tx_v^t$ if $e_4 \neq 0$
- $k$ 2 letter monomials SE to $\varphi(t,t)x_u^tx_v^t$ if $\varphi(t,t) \neq 0$
- $k$ 1 letter monomials SE to $e_3x_u^{2t}$ if $e_3 \neq 0$
- $k$ 1 letter monomials SE to $e_4x_u^{2t}$ if $e_4 \neq 0$

This list incorporates all the ways that one letter monomials of degree $2t$ and two letter monomials that are $\triangleright$-equivalent to $x_u^tx_v^t$ can appear in the given family of polynomials. Moreover, there is no cancellation:

Each of the $k$ polynomials that sit on the diagonal in the $k \times k$ array corresponding to $p(X,Y)$ contains

$$(\chi(e_1) + \chi(e_2) + \chi(e_3) + \chi(e_4))(k - 1) \text{ monomials } \triangleright \text{ to } x_u^tx_v^t$$

made up of off-diagonal letters and

$$\chi(\varphi(t,t)) \text{ monomials } \triangleright \text{ to } x_u^tx_v^t$$

(2.13)

made up of diagonal letters, as well as exactly one one letter monomial of degree $2t$ with coefficient $e_3$ and exactly one one letter monomial of degree $2t$ with coefficient $e_4$, both of which are diagonal entries.

No off diagonal polynomial contains any two letter monomials $\triangleright$ to $x_u^tx_v^t$.

**Proof.** Two letter words $x_u^tx_j^t$ with $i \neq j$ and $t \geq 2$ may be generated in four different ways:

1. as entries in either $m_{\alpha,\beta}(X,Y)$ or $m_{\alpha,\beta}(Y,X)$ with $\alpha$ and $\beta$ subject to (2.1) and (2.2) with $s = t$ if $x_i$ and $x_j$ are in different matrices;
2. as entries in $X^{2t}$ (resp., $Y^{2t}$) if $x_i$ and $x_j$ are both in $X$ (resp., $Y$).
Suppose first that \( x_i \) and \( x_j \) are in different matrices and that (2.9) is in force. Then Lemma 2.2 implies that the scalar multiples of the words \( x_i^t x_j^t \) will appear in at least one of the \( k^2 \) polynomials if and only if \( x_i \) and \( x_j \) are diagonal entries in the same position. In this instance, \( x_i^t x_j^t \) will appear in a polynomial that sits in the same diagonal position as \( x_i \) and \( x_j \) with coefficient \( \varphi(t, t) \).

Suppose next that \( x_i \) and \( x_j \) are in different matrices and that (2.11) is in force. Then, in view of Lemma 2.4, it remains only to consider the contributions from \((XY)^t \) and \((YX)^t \): If
\[
X = x_i E_{ab} + \cdots \quad \text{and} \quad Y = x_j E_{ba} + \cdots,
\]
then
\[
(2.14) \quad (XY)^t = x_i^t x_j^t E_{aa} + \cdots \quad \text{and} \quad (YX)^t = x_i^t x_j^t E_{bb} + \cdots.
\]

Since there are \( k^2 - k \) off-diagonal positions in a \( k \times k \) matrix, there are \( k^2 - k \) choices of \( E_{ab} \) with \( a \neq b \). Moreover, since the entry \( x_i^t x_j^t \) appears in the \( aa \) position in \((XY)^t \) and the \( bb \) position in \((YX)^t \) there will be no cancellation, even if \( e_2 = -e_1 \).

On the other hand contributions that come from diagonal entries of \( X \) and \( Y \) can interact with each other, i.e., if
\[
X = x_i E_{aa} + \cdots \quad \text{and} \quad Y = x_j E_{aa} + \cdots,
\]
then
\[
e_1(XY)^t + e_2(YX)^t = (e_1 + e_2)x_i^t x_j^t E_{aa} + \cdots
\]
and
\[
e_1(XY)^t + e_2(YX)^t + q(X, Y) = \varphi(t, t)x_i^t x_j^t E_{aa} + \cdots.
\]

Thus, if \( \varphi(t, t) \neq 0 \), there will be \( k \) contributions, one for each choice of \( a \in \{1, \ldots, k\} \).

The contributions from \( e_3X^{2t} \) and \( e_4Y^{2t} \) are enumerated in much the same way. Moreover, there is no cancellation, because the monomials with coefficient \( e_3 \) have all their letters in \( X \) and the monomials with coefficient \( e_4 \) have all their letters in \( Y \). □

**Remark 2.9.** The list in Lemma 2.8 is written under the assumption that \( e_1, e_2, e_3 \) and \( e_4 \) are four distinct numbers. If, say, \( e_1, e_2 \) and \( e_3 \) are three distinct numbers and \( e_4 = e_3 \), then there will instead be \( 2(k^2 - k) \) two letter monomials \( SE \) to \( e_3 x_u x_v \), \( k^2 - k \) two letter terms \( SE \) to \( e_1 x_u x_v \), \( k^2 - k \) two letter terms \( SE \) to \( e_2 x_u x_v \), \( k^2 - k \) two letter terms \( SE \) to \( \varphi(t, t)x_u x_v \) and the one letter monomials would be as they are stated above.

**Lemma 2.10.** Let \( p_1, \ldots, p_{2k^2} \) be a family of polynomials in the \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) that admits an nc representation \( p(X, Y) \) of degree \( d > 1 \). Suppose further that \( t \geq 1 \),
\[
(2.15) \quad p(X, Y) = f_1(XY)^t X + f_2(YX)^t Y + f_3 X^{2t+1} + f_4 Y^{2t+1} + q(X, Y),
\]
where \( q \) is a polynomial that does not contain any scalar multiples of the first four monomials listed in (2.15), the coefficients \( f_1, \ldots, f_4 \) are distinct and \( x_u \neq x_v \). Then
the family of \( k^2 \) polynomials will contain exactly

\[
\begin{align*}
\text{k}^2 - k & \quad 2 \text{ letter monomials } SE \ f_1 x_u^{t+1} x_v^t & \quad \text{if } f_1 \neq 0 \\
\text{k}^2 - k & \quad 2 \text{ letter monomials } SE \ f_2 x_u^t x_v^{t+1} & \quad \text{if } f_2 \neq 0 \\
\text{k}^2 - k & \quad 2 \text{ letter monomials } SE \ f_3 x_u^{t+1} x_v^t & \quad \text{if } f_3 \neq 0 \\
\text{k}^2 - k & \quad 2 \text{ letter monomials } SE \ f_4 x_u^t x_v^{t+1} & \quad \text{if } f_4 \neq 0 \\
k & \quad 2 \text{ letter monomials } SE \ \varphi(t + 1, t) x_u^{t+1} x_v^t & \quad \text{if } \varphi(t + 1, t) \neq 0 \\
k & \quad 2 \text{ letter monomials } SE \ \varphi(t, t + 1) x_u^t x_v^{t+1} & \quad \text{if } \varphi(t, t + 1) \neq 0 \\
k & \quad 1 \text{ letter monomials } SE \ f_3 x_u^{2t+1} & \quad \text{if } f_3 \neq 0 \\
k & \quad 1 \text{ letter monomials } SE \ f_4 x_u^{2t+1} & \quad \text{if } f_4 \neq 0
\end{align*}
\]

This list incorporates all the ways that one letter monomials of degree \( 2t + 1 \) and two letter monomials \( \vartriangleright \) to \( x_u^{t+1} x_v^t \) can appear in the given family of polynomials. Moreover, there is no cancellation.

Each of the \( k^2 - k \) polynomials that are off the diagonal in the \( k \times k \) array corresponding to \( p(X, Y) \) contains

\( (\chi(f_1) + \chi(f_2) + \chi(f_3) + \chi(f_4)) \) two letter monomials \( \vartriangleright \) to \( x_u^{t+1} x_v^t \)

made up of off-diagonal letters. Each of the \( k \) polynomials that are on the diagonal contains

\[
(\chi(\varphi(t + 1, t)) + \chi(\varphi(t, t + 1))) \quad \text{two letter monomials } \vartriangleright \ x_u^{t+1} x_v^t
\]

made up of diagonal letters, as well as a one letter monomial \( SE \) to \( f_3 x_u^{2t+1} \) if \( f_3 \neq 0 \) and a one letter monomial \( SE \) to \( f_4 x_u^{2t+1} \) if \( f_4 \neq 0 \). The letters in these one letter monomials are diagonal entries.

**Proof.** Let \( \alpha = (\alpha_1, \ldots, \alpha_\ell) \) and \( (\beta_1, \ldots, \beta_\ell) \) be multi-indices that meet conditions (2.1) and (2.2) and set \( s = t + 1 \). Two letter words \( x_i^{t+1} x_j^t \) with \( i \neq j \) and \( t \geq 2 \) may be generated in four different ways:

1. by \( m_{\alpha, \beta}(X, Y) \) if \( x_i \in X \) and \( x_j \in Y \);
2. by \( m_{\alpha, \beta}(Y, X) \) if \( x_i \in Y \) and \( x_j \in X \);
3. by \( X^{2t+1} \) if \( x_i \in X \) and \( x_j \in Y \); and
4. by \( Y^{2t+1} \) if \( x_i \in Y \) and \( x_j \in X \).

If (2.9) is in force, then the two letter words \( x_i^{t} x_j^{t} \) with \( i \neq j \) can only come from diagonal pairs in the same position. The same terms with possibly different coefficients may appear from the diagonal entries in \( X \) and \( Y \) from polynomials of degree \( t + 1 \) in \( X \) and \( t \) in \( Y \) or degree \( t \) in \( X \) and degree \( t + 1 \) in \( Y \) in \( q(X, Y) \). The coefficients of the net contribution are \( \varphi(t + 1, t) \) and \( \varphi(t, t + 1) \), respectively, and there will be a total of \( k \chi(\varphi(t + 1, t)) \) and \( k \chi(\varphi(t, t + 1)) \) such pairs, one of each sort in each polynomial on the diagonal of the \( k \times k \) array of \( p(X, Y) \).

On the other hand, if (2.11) is in force, then

\[
\alpha_1 = \cdots = \alpha_\ell = 1, \quad \beta_1 = \cdots = \beta_{\ell - 1} = 1, \quad \beta_\ell = 0
\]
Thus, if
\[ X = x_i E_{ab} + \cdots \quad \text{and} \quad Y = x_j E_{cd} + \cdots, \]
then, in view of assertions 2 and 4 of Lemma 2.4, the coefficient of \( x_i^t x_j^t \) in \( m_{\alpha,\beta}(X, Y) \) and \( m_{\alpha,\beta}(Y, X) \) will be nonzero if and only if \( c = b \) and \( d = a \). Correspondingly,
\[ (XY)^t X = x_i^{t+1} x_j^t E_{ab} + \cdots \quad \text{and} \quad (YX)^t Y = x_i^t x_j^{t+1} E_{ba} + \cdots. \]

Since there \( k^2 - k \) off-diagonal positions in a \( k \times k \) matrix, there are \( k^2 - k \) choices of \( E_{ab} \) with \( a \neq b \). Moreover, the entry \( f_1 x_i^{t+1} x_j^t \) can not cancel the entry \( f_2 x_i^t x_j^{t+1} \) even if \( a = b \), since \( x_i \) and \( x_j \) are in different matrices. However, there can be contributions from monomials in \( q(X, Y) \) of degree \( t + 1 \) in \( X \) and \( t \) in \( Y \) or degree \( t \) in \( X \) and \( t + 1 \) in \( Y \).

Similarly, if \( a \neq b \) and \( X = x_i E_{ab} + x_j E_{ba} + \cdots \) (resp., \( Y = x_i E_{ab} + x_j E_{ba} + \cdots \)), then
\[ X^{+t} = x_i^s x_j^t E_{ab} + x_i^t x_j^s E_{ba} + \cdots \quad \text{(resp.,} \quad Y^{+t} = x_i^s x_j^t E_{ab} + x_i^t x_j^s E_{ba} + \cdots). \]
The final assertion comes by counting the contributions discussed above. \( \square \)

**Remark 2.11.** The list in Lemma 2.10 is written under the assumption that \( f_1, f_2, f_3 \) and \( f_4 \) are four distinct numbers. If, say, \( f_1, f_2 \) and \( f_3 \) are three distinct numbers and \( f_4 = f_3 \), then there will instead be \( 2(k^2 - k) \) two letter monomials \( SE \) to \( f_3 x_i^{t+1} x_j^t, \)
\( (k^2 - k) \) two letter terms \( SE \) to \( f_1 x_i^{t+1} x_j^t, \)
\( (k^2 - k) \) two letter terms \( SE \) to \( f_2 x_i^t x_j^{t+1}, \)
\( k \) two letter terms \( SE \) to \( \varphi(t + 1, t) x_i^{t+1} x_j^t \) and the one letter monomials would be as they are stated above.

### 2.3.1. Enumeration of two letter monomials with one letter on the diagonal.

**Lemma 2.12.** Let \( p_1, \ldots, p_{k^2} \) be a family of polynomials in the \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) that admits an nc representation \( p(X, Y) \) of degree \( d > 1 \). Suppose further that \( t \geq 2 \),
\[ p(X, Y) = d_1 X^t + d_2 X^{t-1} Y + d_3 Y X^{t-1} + q(X, Y), \]
where \( q(X, Y) \) does not contain any scalar multiples of the first three monomials in (2.17) and the coefficients \( d_1, \ldots, d_3 \) are distinct.

If \( x_u \) is a diagonal element of \( X \), then the family of \( k^2 \) polynomials will contain exactly
\[
\begin{align*}
2k - 2 & \quad \text{2 letter monomials} \quad d_1 x_u^{t-1} x_v \quad \text{with} \quad x_v \neq x_u \quad \text{if} \quad d_1 \neq 0 \\
2k - 1 & \quad \text{2 letter monomials} \quad d_2 x_v^{t-1} x_u \quad \text{with} \quad x_v \quad \text{an off-diagonal entry of} \quad Y \quad \text{if} \quad d_2 \neq 0 \\
2k - 1 & \quad \text{2 letter monomials} \quad d_3 x_v^{t-1} x_u \quad \text{with} \quad x_v \quad \text{an off-diagonal entry of} \quad Y \quad \text{if} \quad d_3 \neq 0 \\
1 & \quad \text{2 letter monoml} \quad \varphi(t - 1, 1) x_u^{t-1} x_v \quad \text{with} \quad x_v \quad \text{a diagonal entry of} \quad Y \quad \text{if} \quad \varphi(t - 1, 1) \neq 0
\end{align*}
\]
This list incorporates all the ways that two letter monomials of degree \( t \) with \( x_u \) of degree \( t - 1 \) can appear in the given family of polynomials. Moreover, no two monomials in this list are the same.
If \( x_u \) is in the \( aa \) position of \( X \) and \( p_{ab} \) denotes the polynomial in the \( ab \) position in the \( k \times k \) array corresponding to \( p(X,Y) \), then

\[
\begin{align*}
(2.18) \quad p_{ab}(x_1, \ldots, x_{k^2}) &= d_1 x_u^{t-1} x_v + d_2 x_u^{t-1} x_w + \cdots \quad \text{for } a \neq b,
\end{align*}
\]

where \( x_v \) (resp., \( x_w \)) is in the \( ab \) position in \( X \) (resp., \( Y \)) and there are no other two letter monomials of degree \( t \) in \( p_{ab} \) with \( x_u^{t-1} \) as a factor. Similarly,

\[
\begin{align*}
(2.19) \quad p_{ba}(x_1, \ldots, x_{k^2}) &= d_1 x_u^{t-1} x_m + d_2 x_u^{t-1} x_n + \cdots \quad \text{for } a \neq b,
\end{align*}
\]

where \( x_m \) (resp., \( x_n \)) is in the \( ba \) position in \( X \) (resp., \( Y \)) and there are no other two letter monomials of degree \( t \) in \( p_{ba} \) with \( x_u^{t-1} \) as a factor.

**Proof.** It is readily checked with the aid of the calculations in §2.1 that if

1. \( x_u \) is in the \( aa \) position of \( X \), \( x_v \in X \) and \( x_v \neq x_u \), then \( x_u^{t-1} x_v \) is either in the \( ab \) position of \( X^t \) or the \( ba \) position of \( X^t \) for some \( b \neq a \);
2. \( x_u \) is in the \( aa \) position of \( X \) and \( x_v \in Y \), then \( x_u^{t-1} x_v \) is either in the \( ab \) position of \( X^{t-1}Y \) or the \( ba \) position of \( Y^{t-1}X \) for some \( b \neq a \).

The rest of the proof is straightforward counting and is left to the reader. \( \square \)

**Remark 2.13.** Let

\[
(2.20) \quad p(X,Y) = d_1 X^t + d_2 X^{t-1} Y + d_3 Y X^{t-1} + d_4 Y^t + q(X,Y) \quad \text{for some integer } t \geq 2,
\]

where \( q(X,Y) \) does not contain any scalar multiples of the first four monomials in (2.20) and assume that \( d_1, \ldots, d_4 \) are subject to the constraints

\[
(2.21) \quad d_1 \neq 0, \quad d_1 \neq d_4 \quad \text{and} \quad d_1 \neq d_2 \quad \text{or} \quad d_1 \neq d_3.
\]

Then there will be \( k \) terms

\[
d_1 x_{i_1}^t, \ldots, d_1 x_{i_k}^t
\]

in the family of \( k^2 \) polynomials and the corresponding letters \( x_{i_1}^t, \ldots, x_{i_k}^t \) may be identified as the diagonal elements of say \( X \). The assumption \( d_1 \neq d_4 \) insures that they can be chosen unambiguously. Thus, if \( x_u \) is one of these diagonal elements and it is in the \( aa \) position of \( X \), then

\[
\begin{align*}
p_{aa} &= d_1 x_u^t + \varphi(t-1,1) x_u^{t-1} x_h + \cdots, \\
p_{ab} &= d_1 x_u^{t-1} x_v + d_2 x_u^{t-1} x_f + \cdots, \\
p_{ba} &= d_1 x_u^{t-1} x_z + d_3 x_u^{t-1} x_g + \cdots,
\end{align*}
\]

where \( x_h \) is in the \( aa \) position of \( Y \), \( x_v \) is in the \( ab \) position of \( X \), \( x_f \) is in the \( ab \) position of \( Y \), \( x_z \) is in the \( ba \) position of \( X \) and \( x_g \) is in the \( ba \) position of \( Y \).

### 2.4. Determining Diagonal Elements.

Lemmas 2.8 and 2.10 serve to enumerate the diagonal entries in \( X \) and \( Y \) when the given set of \( k^2 \) polynomials contain one letter monomials. But if say \( e_3 = e_4 \) in Lemma 2.8 and \( f_3 = f_4 \) in Lemma 2.10, then it is not immediately obvious which entries belong to \( X \) and which entries belong to \( Y \).
Definition 2.14. A pair of variables $x_i$ and $x_j$ will be called a partitioned (resp., dyslexic) diagonal pair if both occur in the $aa$ position for some choice of $a \in \{1, \ldots, k\}$ and we know (resp., do not know) which variable occurs in $X$ and which occurs in $Y$.

Lemma 2.15. Suppose $p_1, \ldots, p_{2k^2}$ is a family of polynomials in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ that admits an nc representation $p(X,Y)$ of degree $d > 1$ such that at least one of the given polynomials contains a term of the form $ex_i^s m(x_1, \ldots, x_{2k^2})$, where $e \in \mathbb{R} \setminus \{0\}$ and $m(x_1, \ldots, x_{2k^2})$ is a monomial of degree $t \geq 1$ that does not contain any $x_i$ terms and $s \geq t+2$. Then $x_i$ lies on the diagonal of either $X$ or $Y$.

Proof. For the sake of definiteness, assume $x_i \in X$. Then, since $s \geq t+2$, every permutation of the symbols $X^s Y^t$ must contain at least two adjacent $X$’s. But if $X = x_i E_{cd} + \cdots$ with $c \neq d$, then $X^2 = 0$. Thus, the given family of polynomials will only contain terms of the form $ex_i^s m(x_1, \ldots, x_{2k^2})$ if $c = d$, i.e., if $x_i$ is a diagonal element of $X$. □

Lemma 2.16. Let $p_1, \ldots, p_{k^2}$ be a family of polynomials in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ that admits an nc representation $p(X,Y)$ of degree $d > 1$. Suppose further that at least one of the polynomials contains at least one term of the form $ex_i^s x_j^t$ with $s \geq t+2$, and $t \geq 2$. Then

1. $x_i$ and $x_j$ are a dyslexic diagonal pair.

2. There exist exactly $k$ dyslexic diagonal pairs $\{x_{i_1}, x_{j_1}\}, \ldots, \{x_{i_k}, x_{j_k}\}$ and $k$ polynomials $p_{\ell_1}, \ldots, p_{\ell_k}$ such that

$$p_{\ell_n} = ex_{i_n}^s x_{j_n}^t + \cdots$$

for $n = 1, \ldots, k$. Moreover, $p_{\ell_n}$ occupies the same diagonal position as $x_{i_n}$ and $x_{j_n}$.

Proof. If $x_i \in X$, then (since $s \geq t+2$ and $t \geq 2$) Lemma 2.15 guarantees that $x_i$ lies on the diagonal of $X$ and that $x_j$ lies on the diagonal of the matrix that contains $x_j$. Since $x_i$ and $x_j$ must be in the same diagonal position, this forces $x_j$ to belong to $Y$. Similarly, if $x_i \in Y$, then $x_j$ must be in $X$ and both variables must be in the same diagonal position. Thus (1) holds; (2) follows automatically from (1), since all the dyslexic diagonal pairs are subject to the same constraints. □

Lemma 2.17. Let $p_1, \ldots, p_{k^2}$ be a family of polynomials in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ that admits an nc representation $p(X,Y)$ of degree $d > 1$. If $t \geq 2$ and $s = t+1$ or $s = t$ and it is also assumed that

1. $k \geq 3$ (so that $k^2 - k > k$) and

2. there exist exactly $k$ monomials $ex_{i_1}^s x_{j_1}^t, \ldots, ex_{i_k}^s x_{j_k}^t$ in the given family of polynomials that are structurally equivalent to $ex_u^s x_v^t$ with $x_u \neq x_v$ and $e \neq 0$,

then $x_{i_n}$ and $x_{j_n}$ are a dyslexic diagonal pair for each $1 \leq n \leq k$. Moreover, if $p_{\ell_n} = ex_{i_n}^s x_{j_n}^t + \cdots$, then $p_{\ell_n}$ occupies the same diagonal position as $x_{i_n}$ and $x_{j_n}$.

Proof. Since there are only $k$ terms in this list and $k^2 - k > k$ when $k \geq 3$, the conclusion follows from Lemma 2.8 if $s = t$ and from Lemma 2.10 if $s = t+1$. In
Lemma 2.18. Suppose that the term \( ax_i^s x_j^t, a \neq 0 \), occurs in some polynomial \( p \) in the given family. If the pair \( x_i \) and \( x_j \) is a dyslexic diagonal pair, then the polynomial is a diagonal polynomial that occurs in the same diagonal position as the dyslexic pair.

Proof. If, say, \( X = x_i E_{aa} + \cdots \), and \( Y = x_j E_{aa} + \cdots \) and the multi-indices \( \alpha \) and \( \beta \) are subject to (2.1) and (2.2), then \( m_{\alpha, \beta}(X, Y) = x_i^s x_j^t E_{aa} + \cdots \). □

Lemma 2.19. Let \( p_1, \cdots, p_{2k^2} \) be a family of polynomials in the \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) that admits an nc representation \( p(X, Y) \) of degree \( d \) with \( d > 1 \) such that exactly \( k \) SE terms of the form \( ax_i^s x_j^t, \cdots, ax_i^s x_j^t \) appear in the family with \( s > t, t \geq 2 \) and \( a \neq 0 \). If \( \{ x_i, x_j \}, \ldots, \{ x_{ik}, x_{jk} \} \) are dyslexic diagonal pairs, then the variables \( x_{i1}, \cdots, x_{ik} \) of degree \( s \) in the terms \( ax_{i1}^s x_{j1}^t, \cdots, ax_{ik}^s x_{jk}^t \) are the diagonal elements of one matrix and and the variables \( x_{j1}, \cdots, x_{jk} \) of degree \( t \) are the diagonal elements of the other matrix.

Proof. If the lemma is false, then without loss of generality, we may suppose that \( x_{i1} \in X \) and \( x_{i2} \in Y \). By Lemmas 2.16 and 2.10, \( \varphi(s, t) = \varphi(t, s) = a \neq 0 \). Thus, as \( s > t \), each diagonal pair \( x_{in}, x_{jn} \) will occur in the monomials \( ax_{in}^s x_{jn}^t \) and \( ax_{in}^t x_{jn}^s \). Therefore, there will be \( 2k \) terms structurally equivalent to \( ax_{i1}^s x_{j1}^t \), which contradicts one of the given assumptions. □

3. Partitioning Algorithms for Families Containing Single Letter Monomials \( ax_i^n, n \geq 2 \)

In the previous section we developed a number of methods to determine the diagonal variables for a family \( \mathcal{P} \) of polynomials \( p_1, \cdots, p_{k^2} \) with an nc representation. The next step is to utilize this information to determine which of the commutative variables \( x_1, \cdots, x_{2k^2} \) are entries in \( X \) and which are entries in \( Y \). In this section we develop two algorithms that we will refer to as partitioning algorithms, since they partition the commutative variables between the two matrices. First, however, we review some preliminary calculations that will be essential in the development of the first partitioning algorithm.

The following assumptions will be in force for the rest of this section:

(A1) \( p_1, \ldots, p_{k^2} \) is a family of polynomials in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) that admits an nc representation \( p(X, Y) \) of degree \( d \) with \( d > 1 \).

(A2) \( \varphi(n, 0) \neq 0 \) for some \( n \geq 2 \).

(A3) \( p(0, 0) = 0 \). (This involves no real loss of generality, because constant terms can be reinserted at the end.)

3.1. Preliminary calculations for the partitioning algorithm. First observe that

\[
(\alpha I_k + \beta E_{st})^n = \alpha^n I_k + n \alpha^{n-1} \beta E_{st} \quad \text{if } s \neq t.
\]
Then without loss of generality, assume that the variables \( \{x_{i_1}, \ldots, x_{i_k}\} \) are the commutative variables that occur on the diagonal of \( X \).

Next, choose a variable \( x_j \) such that
\[
x_j \notin \{x_{i_1}, \ldots, x_{i_k}\},
\]
set
\[
x_{i_1} = \cdots = x_{i_k} = \alpha \quad \text{and} \quad x_j = \beta
\]
and set all the other variables equal to zero.

Then there are three mutually exclusive possibilities:

1. \( x_j \in X \). In this case
\[
X = \alpha I_k + \beta E_{st} \quad \text{with} \quad s \neq t \quad \text{and} \quad Y = 0.
\]

2. \( x_j \in Y \) but is not on the diagonal of \( Y \). In this case
\[
X = \alpha I_k \quad \text{and} \quad Y = \beta E_{st} \quad \text{with} \quad s \neq t.
\]

3. \( x_j \in Y \) and is on the diagonal of \( Y \). In this case
\[
X = \alpha I_k \quad \text{and} \quad Y = \beta E_{ss} \quad \text{for some integer} \quad s \in \{1, \ldots, k\}.
\]

All three of these cases fit into the common framework of choosing \( X = A \in \mathbb{R}^{k \times k} \) and \( Y = B \in \mathbb{R}^{k \times k} \) with \( AB = BA \). Thus, if
\[
p(X, Y) = \sum_{i=0}^{d} p_i(X, Y),
\]
where \( p_i(X, Y) \) denotes the terms in \( p(X, Y) \) of degree \( i \), then the condition \( AB = BA \) insures that
\[
(3.1) \quad p_i(A, B) = \sum_{j=0}^{i} c_{i-j,j} A^{i-j} B^j = c_{i0} A^i + \sum_{j=1}^{i} c_{i-j,j} A^{i-j} B^j = \hat{p}_i(A, B),
\]
where
\[
c_{ij} = \varphi(i, j) \quad \text{(for short)} \quad \text{for} \quad i, j = 0, \ldots, d \quad \text{with} \quad c_{ij} = 0 \quad \text{for} \quad i + j > d
\]
and hence,
\[
(3.2) \quad p(A, B) = \sum_{i=1}^{d} \sum_{j=0}^{i} c_{i-j,j} A^{i-j} B^j = \sum_{i=1}^{d} c_{i0} A^i + \sum_{i=1}^{d} \sum_{j=1}^{i} c_{i-j,j} A^{i-j} B^j = \hat{p}(A, B),
\]
since the assumption \( p(0, 0) = 0 \) forces \( c_{00} = 0 \).

In Case 1, \( A = \alpha I_k + \beta E_{st} \) with \( s \neq t \) and \( B = 0 \). Therefore,
\[
(3.3) \quad p_i(A, B) = c_{i0}(\alpha^i I_k + i \alpha^{i-1} \beta E_{st}) \quad \text{for} \quad i \geq 1 \quad \text{and}
\]
\[ p(A, B) = \sum_{i=1}^{d} c_{i0} \alpha^i I_k + \sum_{i=1}^{d} c_{i0} i \alpha^{i-1} \beta E_{st} \]

(3.4)

\[ = \sum_{i=1}^{d} c_{i0} \alpha^i I_k + \sum_{i=0}^{d-1} c_{i+1,0} (i + 1) \alpha^i \beta E_{st}. \]

In Case 2, \( A = \alpha I_k \) and \( B = \beta E_{st} \) with \( s \neq t \). Therefore,

\[ p_{ij}(A, B) = c_{i0} \alpha^i I_k + c_{i-1,1} \alpha^{i-1} \beta E_{st} \quad \text{for } i \geq 1 \quad \text{and} \]

(3.5)

\[ p(A, B) = \sum_{i=1}^{d} c_{i0} \alpha^i I_k + \sum_{i=1}^{d} c_{i-1,1} \alpha^{i-1} \beta E_{st} \]

(3.6)

In Case 3, \( A = \alpha I_k \) and \( B = \beta E_{ss} \). Therefore,

\[ p_{ij}(A, B) = c_{i0} \alpha^i I_k + \sum_{j=1}^{i} c_{i-j,j} \alpha^{i-j} \beta^j E_{ss} \quad \text{for } i \geq 1 \quad \text{and} \]

(3.7)

\[ p(\alpha I_k, \beta E_{ss}) = \sum_{i=1}^{d} c_{i0} \alpha^i I_k + \sum_{i=1}^{d} \sum_{j=1}^{i} c_{i-j,j} \alpha^{i-j} \beta^j E_{ss}. \]

(3.8)

We remark that the formula \( p(\alpha I_k + \beta E_{st}, 0) \) in Case 1, can also be expressed in terms of the polynomial

\[ \varphi(\alpha) = \sum_{i=1}^{d} c_{i0} \alpha^i \]

as

\[ p(\alpha I_k + \beta E_{st}, 0) = \varphi(\alpha) I_k + \varphi'(\alpha) \beta E_{st}. \]

The equality \( c_{d0} = a \neq 0 \) guarantees that \( \varphi(\alpha) \neq 0 \) and \( \varphi'(\alpha) \neq 0 \).

3.2. Partitioning Algorithm I. Now we develop the fundamental ideas for an algorithm that will partition the commutative variables between \( X \) and \( Y \).

Set the diagonal entries of \( X \) equal to \( \alpha \), one of the other \( 2k^2 - k \) variables \( x_i \) equal to \( \beta \) and the remaining \( 2k^2 - k - 1 \) variables equal to zero. Then:

\( x_i \) is an off-diagonal entry of \( X \) \iff this substitution produces

\( k \) polynomials equal to \( \sum_{i=1}^{d} c_{i0} \alpha^i \),

\( 1 \) polynomial equal to \( \sum_{i=0}^{d-1} c_{i+1,0} (i + 1) \alpha^i \beta \) and

\( k^2 - k - 1 \) polynomials equal to 0;
is an off-diagonal entry of $Y$ $\iff$ this substitution produces

- $k$ polynomials equal to $\sum_{i=1}^{d} c_{i0}\alpha^i$,
- $1$ polynomial equal to $\sum_{i=0}^{d-1} c_{i1}\alpha^i\beta$ and
- $k^2 - k - 1$ polynomials equal to $0$;

is a diagonal entry of $Y$ $\iff$ this substitution produces

- $k - 1$ polynomials equal to $\sum_{i=1}^{d} c_{i0}\alpha^i$,
- $1$ polynomial equal to $\sum_{i=1}^{d} c_{i0}\alpha^i + \sum_{i=1}^{d} \sum_{j=1}^{i} c_{i-j,j}\alpha^{i-j}\beta^j$ and
- $k^2 - k$ polynomials equal to $0$;

The first two cases will be indistinguishable if and only if

$$\sum_{i=0}^{d-1} c_{i+1,0}(i + 1)\alpha^i\beta = \sum_{i=0}^{d-1} c_{i1}\alpha^i\beta$$

for every choice of $\alpha$ and $\beta$, i.e., if and only if

$$c_{i+1,0}(i + 1) = c_{i1} \quad \text{for } i = 0, \ldots, d - 1.$$

### 3.3. Partitioning with homogeneous components: Algorithm DiagPar1

It is often advantageous to focus on the homogeneous components of the given set of polynomials $p_1, \ldots, p_{k^2}$, i.e., on the sub-polynomials of $p_1, \ldots, p_{k^2}$ of specified degree. It is readily seen that if $p_{[n]}(X,Y)$ denotes the terms in the nc polynomial of degree $n$, then the $k^2$ commuting polynomials in the $k \times k$ array corresponding to $p_{[n]}(X,Y)$ are the $k^2$ polynomials $q_1, \ldots, q_{k^2}$, where $q_i(x_1, \ldots, x_{2k^2})$ is the sum of the monomials in $p_i(x_1, \ldots, x_{2k^2})$ of degree $n$.

The three possibilities considered earlier applied to polynomials of degree $n$ lead to simpler criteria:

1. If $A = \alpha I_k + \beta E_{st}$ with $s \neq t$ and $B = 0$, then

$$p_{[n]}(A, B) = c_{n0}(\alpha^n I_k + n\alpha^{n-1}\beta E_{st}).$$

2. If $A = \alpha I_k$ and $B = \beta E_{st}$ with $\beta \neq 0$ and $s \neq t$, then

$$p_{[n]}(A, B) = c_{n0}\alpha^n I_k + c_{n-1,1}\alpha^{n-1}\beta E_{st}.$$
3. If $A = \alpha I_k$ and $B = \beta E_{ss}$, then,

$$p_{[n]}(A,B) = c_{n0}\alpha^n I_k + \sum_{j=1}^{n} c_{n-j,j} \alpha^{n-j} \beta^j E_{ss}.$$ 

Thus, if the diagonal entries of $X$ are set equal to $\alpha$, one of the other $2k^2-k$ variables $x_i$ is set equal to $\beta$ and the remaining $2k^2-k-1$ variables are set equal to zero, and if $\beta \neq \alpha$ and $\varphi(n,0) \neq 0$, then:

- $x_i$ is an off-diagonal entry of $X \iff$ this substitution produces $k$ polynomials equal to $c_{n0}\alpha^n$, $1$ polynomial equal to $c_{n0}n\alpha^{n-1}\beta$, and $k^2-k-1$ polynomials equal to $0$;

- $x_i$ is an off-diagonal entry of $Y \iff$ this substitution produces $k$ polynomials equal to $c_{n0}\alpha^n$, $1$ polynomial equal to $c_{n-1,1}\alpha^{n-1}\beta$, and $k^2-k-1$ polynomials equal to $0$;

- $x_i$ is a diagonal entry of $Y \iff$ this substitution produces $k-1$ polynomials equal to $c_{n0}\alpha^n$, $1$ polynomial equal to $c_{n0}\alpha^n + \sum_{j=1}^{n} c_{n-j,j} \alpha^{n-j} \beta^j$ and $k^2-k$ polynomials equal to $0$.

(1) The first two possibilities will be distinguishable if and only if $n\varphi(n,0) \neq \varphi(n-1,1)$.

(2) The third will be distinguishable from the first if $\varphi(n,0) \neq 0$ by counting the number of polynomials equal to zero, $k+1$ vs $k$.

(3) The third will be distinguishable from the second if and at least two of the coefficients $\varphi(n-j,j) \neq 0$ for $j = 0, 1, \ldots, n$ are nonzero. This is done by comparing number of terms of polynomials.

We will refer to the process developed in the previous discussion as **Algorithm DiagPar1**. In the following theorem we summarize the conditions under which this algorithm will partition the commutative variables. The reader should keep in mind that if there are $k$ single letter monomials (as opposed to $2k$ or $0$) of degree $n$ in a homogeneous family $P$ of polynomials of degree $n$, then we always assume that the associated variables lie on the diagonal of $X$ and hence that $\varphi(n,0) \neq 0$.

**Theorem 3.1** (DiagPar1 Algorithm). Let $p_1, \cdots, p_k$ be a family of homogeneous polynomials with an nc representation $p(X,Y)$ of degree $n$ with $n \geq 2$. Then Algorithm
DiagPar1 successfully partitions the variables $x_1, \cdots, x_{2k^2}$ between $X$ and $Y$ if and only if
\begin{equation}
\varphi(n, 0) \neq 0, \quad \varphi(n, 0) \neq \varphi(0, n), \quad \text{and} \quad n\varphi(n, 0) \neq \varphi(n - 1, 1).
\end{equation}

**Proof.** If $\varphi(n, 0) \neq 0$ and (3.9) holds, then the above discussion implies that DiagPar1 will successfully partition the variables between $X$ and $Y$.

Conversely, suppose that DiagPar1 successfully partitions the variables between $X$ and $Y$. This implies that the algorithm can first determine the set of diagonal variables of $X$ by analyzing single letter monomials of the form $ax^n$ in the polynomials in $P$. By Lemma 2.6, we see that this is possible only if $\varphi(n, 0) \neq 0$ and $\varphi(n, 0) \neq \varphi(0, n)$. Finally, the above discussion implies that DiagPar1 partitions the off-diagonal elements between $X$ and $Y$ if and only if $\varphi(n, 0) \neq \varphi(n - 1, 1)$. \qed

### 3.4. Partitioning: Algorithm DiagPar2.

In this subsection we develop another partitioning algorithm that is closely related to the DiagPar1 Algorithm and is based on the following observation: Let $p_1, \cdots, p_{2k^2}$ be a family of polynomials with an nc representation $p(X,Y)$ and suppose that for some $t \geq 2, \varphi(t, 0) \neq 0$ and $\varphi(t, 0) \neq \varphi(0, t)$. Then by Lemma 2.6, there exists exactly $k$ polynomials $p_{i_1}, \cdots, p_{i_k}$, each of which contains exactly one one letter monomial of degree $t$ with coefficient $\varphi(t, 0)$.

The next step rests on Lemma 2.12. But for ease of understanding, let $x_{ij}$ denote the $ij$ entry in $X, y_{ij}$ the $ij$ entry in $Y$ and let $p_{ij}$ denote the $ij$ entry in an array of commutative polynomials that admits an nc representation $p(X,Y)$. Then
\[ p_{ii} = \varphi(t, 0)x_{ii}^t + \varphi(0, t)y_{ii}^t + \cdots, \]
whereas for $i \neq j$,
\begin{align*}
p_{ij} &= \varphi(t, 0)(x_{ii}^{t-1}x_{ij} + x_{ij}x_{jj}^{t-1}) + \varphi(0, t)(y_{ii}^{t-1}y_{ij} + y_{ij}y_{jj}^{t-1}) \\
&\quad + \varphi(Y; t - 1, 1)y_{ij}x_{ii}^{t-1} + \varphi(t - 1, 1, Y)x_{ii}^{t-1}y_{ij} + \cdots
\end{align*}
and
\begin{align*}
p_{ji} &= \varphi(t, 0)(x_{ji}x_{ii}^{t-1} + x_{jj}^{t-1}x_{ji}) + \varphi(0, t)(y_{ji}y_{ii}^{t-1} + y_{ij}y_{jj}^{t-1}) \\
&\quad + \varphi(Y; t - 1, 1)y_{ji}x_{ii}^{t-1} + \varphi(t - 1, 1, Y)x_{jj}^{t-1}y_{ji} + \cdots
\end{align*}

Thus, if the entry $x_{ii}$ and $t$ are known, then there will be exactly two two letter monomials in $p_{ij}$, $i \neq j$ with $x_{ii}^{t-1}$ as a factor, namely, $\varphi(t, 0)x_{ji}x_{ii}^{t-1}$ and $\varphi(t - 1, 1, Y)x_{ii}^{t-1}y_{ij}$, but only the first of these will have the correct coefficient if $\varphi(t, 0) \neq \varphi(t - 1, 1, Y)$.

Thus, under this condition, it is possible to isolate all the entries in $X$ by repeating the argument for $i = 1, \ldots, k$.

Similar considerations based on inspection of the polynomials $p_{ji}$ will also yield all the entries in $X$ if $\varphi(t, 0) \neq \varphi(Y; t - 1, 1)$.

We will refer to the above procedure for partitioning the commutative variables as Algorithm DiagPar2. The conditions under which this algorithm works are summarized in the following theorem.

**Theorem 3.2** (DiagPar2 Algorithm). Let $p_1, \cdots, p_{2k^2}$ be a family of polynomials with an nc representation $p(X,Y)$. Then Algorithm DiagPar2 will partition the variables
between \( X \) and \( Y \) if and only if there exists an \( n \geq 2 \) such that \( \varphi(n, 0) \neq 0 \), \( \varphi(n, 0) \neq \varphi(0, n) \), and
\[
(3.10) \quad \varphi(n, 0) \neq \varphi(n - 1, 1; Y) \quad \text{or} \quad \varphi(n, 0) \neq \varphi(Y; n - 1, 1).
\]

Proof. If the above conditions hold then the above discussion implies that Algorithm DiagPar2 will successfully partition the variables between \( X \) and \( Y \). Conversely, if DiagPar2 partitions the variables between \( X \) and \( Y \), then it must first determine the diagonal elements. Lemma 2.6 implies that this is only possible if \( \varphi(n, 0) \neq 0 \) and \( \varphi(n, 0) \neq \varphi(0, n) \). The above discussion implies that DiagPar2 will successfully partition the off-diagonal entries only if the above conditions hold. \( \square \)

3.5. Summary of Partitioning Algorithms. The main conclusion of this section is that if the given system \( P \) of \( k^2 \) commutative polynomials admits an nc representation \( p(X, Y) \) in the set \( NC^{(3.11)} \) of nc polynomials \( p(X, Y) \) of degree \( d > 1 \) for which there exists an integer \( n \geq 2 \) such that
\[
(3.11) \quad \begin{cases} 
\varphi(n, 0) \neq 0, & \varphi(n, 0) \neq \varphi(0, n), \quad \text{and either} \\
n \varphi(n, 0) \neq \varphi(n - 1, 1) \quad \text{or} \\
\varphi(n, 0) \neq \varphi(n - 1, 1; Y) \quad \text{or} \quad \varphi(n, 0) \neq \varphi(Y; n - 1, 1),
\end{cases}
\]
then either Algorithm DiagPar1 or Algorithm DiagPar2 will successfully partition the variables between \( X \) and \( Y \).

4. Positioning Algorithms for Families of Polynomials Containing One Letter Monomials

In this section we shall present algorithms for positioning the variables in \( X \), given that the diagonal entries of \( X \) are known and that the remaining \( 2k^2 - k \) commutative variables are partitioned between \( X \) and \( Y \). The assumptions (A1), (A3) that are listed at the beginning of §3 and a weaker form of (A2):
\[
(A2') \quad |\varphi(n, 0)| + |\varphi(0, n)| > 0 \text{ for some } n \geq 2,
\]
will be in force for the rest of this section.

4.1. Positioning the variables within \( X \): Algorithm ParPosX. If we can determine the diagonal variables of the matrix \( X \) and implement Algorithm DiagPar1 or DiagPar2 in §3, then we may assume that \( x_{i1}, \ldots, x_{i_k} \in X \), and the remaining \( k^2 \) variables belong to \( Y \). To ease the notation, assume that \( x_1, \ldots, x_{k^2} \in X \) and that \( x_i \) is in the \( ii \) position for \( i = 1, \ldots, k \) and let
\[
\begin{align*}
R_i & = \text{denote the remaining } k - 1 \text{ entries in the } i\text{th row of } X, \\
C_i & = \text{denote the remaining } k - 1 \text{ entries in the } i\text{th column of } X, \quad \text{and} \\
L_i & = R_i \cup C_i.
\end{align*}
\]
Then, since
\[
(4.1) \quad X^n = (x_i^{n-1}E_{ii} + \cdots)(x_jE_{st} + \cdots) = (x_i^{n-1}x_jE_{ii}E_{st} + \cdots)
\]
and
\[(4.2) \quad X^n = (x_jE_{st} + \cdots)(a_i^{n-1}E_{ii} + \cdots) = (x_i^{n-1}x_jE_{st}E_{ii} + \cdots)\]
for \(1 \leq i \leq k < j\), it is readily seen that the term \(ax_i^{n-1}x_j\) appears in one of the entries of \(aX^n\) if and only if either \(s = i\) or \(t = i\), i.e., if and only if \(x_j \in R_i \cup C_i\). Moreover, since \(R_i \cap C_i = \emptyset\) there will be \(2k - 2\) such terms in \(X^n\).

Let
\[L_i \cap L_r = \{x_j, x_t\} \quad \text{for some integer} \ r \in \{1, \ldots, k\} \setminus \{i\}.\]
Then one of these two variables will be in the \(ir\) position of \(X\), while the other is in the \(ri\) position, and it is impossible to decide which is where without extra information. Let us assume for the sake of definiteness that \(x_j\) is in the \(ir\) position of \(X\), then \(x_t\) will be in the \(ri\) position. Moreover, since
\[X^n = (x_jE_{ir} + \cdots)(a_r^{n-1}E_{rr} + \cdots) = (x_ja_r^{n-1}E_{ir} + \cdots),\]
the term \(x_ja_r^{n-1}\) also belongs to the same polynomial. Thus,
\[X^n = \begin{bmatrix} q_{11} & \cdots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{k1} & \cdots & q_{kk} \end{bmatrix},\]
where the \(q_{st}\) are either homogeneous polynomials of degree \(n\) in the variables \(x_1, \ldots, x_{k^2}\) or zero. In particular, if \(k \geq 3\), \(n \geq 2\) and \(i \neq r\), then
\[q_{ir} = (x_i^{n-1}x_j + x_ja_r^{n-1} + x_i^{n-2}x_tx_m + \cdots)\]
with \(x_t\) and \(x_m\) off the diagonal and \(x_t \neq x_m\). This automatically insures that \(x_t\) and \(x_m\) differ from \(x_i\) and \(x_j\), i.e.,
\[\{x_t, x_m\} \cap \{x_i, x_j\} = \emptyset,\]
and that there exists an integer \(j \in \{1, \ldots, k\} \setminus \{i, r\}\) such that
\[
\begin{align*}
\text{either} \quad & x_t \in R_i \cap C_j \quad \text{and} \quad x_m \in R_j \cap C_r \quad \text{or vice versa}. \\
\end{align*}
\]
Therefore, since \(R_i \cap R_j = \emptyset\) for \(j \neq i\), only one of the two variables \(x_t, x_m\) is listed in the set \(R_i \cup C_i\) and hence these two variables can be positioned unambiguously. This procedure adds one more variable to each of \(R_i\) and \(C_r\). The remaining entries in \(R_i\) and \(C_r\) are obtained by repeating this procedure \(k - 3\) more times by running through all the other triples of the form \(x_i^{n-2}x_ux_v\) in \(q_{ir}\).

After \(R_i\) and \(C_r\) are filled in, the procedure is repeated with some other diagonal element as a starting point, and then repeated again and again until all the \(k^2\) variables are positioned in \(X\).

We will refer to the procedure for positioning the variables outlined in the previous discussion as \textbf{Algorithm ParPosX}. The conditions under which this algorithm will position the commutative variables are summarized in the following theorem.

\textbf{Theorem 4.1.} [Algorithm ParPosX] Let \(p_1, \ldots, p_{k^2}\) be a family of polynomials with an nc representation \(p(X, Y)\). Then Algorithm ParPosX will position the variables in \(X\) if and only if \(p \in NC(3,11)\).
4.2. Algorithm for positioning the polynomials given positioning in $X$. If $X$ and $Y$ are general matrices containing the variables $x_1, \cdots, x_{2k^2}$, an nc representation $p(X,Y)$ produces a family of polynomials $p_1, \cdots, p_{k^2}$ that is arranged in a $k \times k$ matrix. The purpose of this section is to investigate when it is possible to determine the position of the polynomials $p_1, \cdots, p_{k^2}$ in the resulting $k \times k$ matrix. We start with a lemma that will be useful in developing a procedure to accomplish this task.

Lemma 4.2. Let $p_1, \cdots, p_{k^2}$ be a homogeneous family of polynomials with an nc representation $p(X,Y)$ of degree $d > 1$. Suppose that the variables $x_1, \cdots, x_{2k^2}$ have been partitioned between $X$ and $Y$ and positioned in $X$. Then, if $x_i$ is in the $aa$ position of $X$, $x_j$ is in the $ab$ position of $X$, $x_\ell$ is in the $bb$ position of $X$ and

$$\varphi(n,0) \neq 0 \quad \text{for some } n \geq 2,$$

then there exists exactly one polynomial that contains the monomials $\varphi(n,0)x_i^{n-1}x_j$ and $\varphi(n,0)x_jx_\ell^{n-1}$. Moreover, this polynomial is in the $ab$ position in the $k \times k$ array.

Proof. This is an easy consequence of the discussion in Subsection 4.1. □

Lemma 4.2 suggests a very simple algorithm for positioning the polynomials in our family. If the commutative variables are partitioned and positioned in the matrix $X$ and if $\varphi(n,0) \neq 0$ for some $n \geq 2$, we simply run through the terms $\varphi(n,0)x_i^{n-1}x_j$ for each $x_i$ in the $ii$ position in $X$ and $x_j$ in the $ij$ position in $X$. The polynomial that contains this monomial will be in the $ij$ position in the array of polynomials. We will refer to this procedure as Algorithm PosPol. The next proposition summarizes the conditions under which it is applicable.

Proposition 4.3 (Algorithm PosPol). Let $p_1, \cdots, p_{k^2}$ be a homogeneous family of polynomials with an nc representation $p(X,Y)$ of degree $d > 1$. Then Algorithm PosPol will successfully position the polynomials $p_1, \cdots, p_{k^2}$ if and only if $p \in NC(3,11)$. □

4.3. Positioning $Y$ given the position of the polynomials: Algorithm PosY. Given a family $\mathcal{P}$ of polynomials $p_1, \cdots, p_{k^2}$ in the variables $x_1, \cdots, x_{2k^2}$ with an nc representation $p(X,Y)$, we have to this point developed algorithms to partition the variables between $X$ and $Y$, position the variables in $X$ and position the polynomials $p_1, \cdots, p_{k^2}$. The next step is to develop an algorithm that positions the commutative variables in $Y$.

Suppose that the commutative variables $x_1, \cdots, x_{2k^2}$ have been partitioned between $X$ and $Y$ and positioned in $X$. Furthermore, suppose that the polynomials in the family $\mathcal{P}$ have been positioned and that $\varphi(s,t) \neq 0$ for some $s \geq 0$ and $t \geq 1$. Let $p_{ij}$
denote the polynomial in the \(ij\) position in \(p(X, Y)\). Let \(x_1, \ldots, x_{k^2}\) be the variables contained in \(X\) and let \(x_1, \ldots, x_k\) be the diagonal variables of \(X\). Set \(x_1 = \cdots = x_k = 1\) and \(x_{k+1} = \cdots = x_{k^2} = 0\). On the nc level this is equivalent to setting \(X = I_k\). Then consider the polynomials \(\hat{p}_{ij} = p_{ij}(1, \cdots, 1, 0, \cdots, 0, x_{k+1}, \cdots, x_{k^2})\) in \(k^2\) commuting variables. If \(\mathcal{P}\) has an nc representation \(p(X, Y)\), this collection of polynomials will have an nc representation \(p(I, Y) = p(Y)\) that contains the monomial \(\varphi(s, t)Y^t \neq 0\). Therefore, each diagonal polynomial \(\hat{p}_{ii}\) will contain a monomial of the form \(\varphi(s, t)x_{ji}^t\), where \(j_i \geq k^2 + 1\). This implies that \(x_{ji}\) must be the \(ii\) entry of \(Y\). By repeating this argument for each \(i\) we can position the other diagonal elements of \(Y\). To position the remaining variables, observe that a monomial of the form \(\varphi(s, t)x_{ji}^{t-1}x_u\) will appear in each \(\hat{p}_{ij}\) where \(u \geq k^2 + 1\). Given that \(x_{ji}\) lies in the \(ii\) position of \(Y\) and \(\hat{p}_{ij}\) is in the \(ij\) position in \(p(Y)\), it follows that \(x_u\) is in the \(ij\) position in \(Y\). By repeating this argument for each \(ij\) we can position the other non-diagonal entries of \(Y\).

We will refer to the process described above as \textbf{Algorithm PosY}. We summarize the conditions under which it will successfully position the variables in \(Y\) in the following proposition.

**Proposition 4.4.** [Algorithm PosY] Let \(p_1, \ldots, p_{k^2}\) be a family of polynomials with an nc representation \(p(X, Y)\). Then \textbf{Algorithm PosY} will successfully position the commutative variables in \(Y\) if and only if \(p \in NC_{(3,11)}\) and there exists a pair of integers \(s \geq 0\) and \(t \geq 1\) such that \(\varphi(s, t) \neq 0\).

**Proof.** This follows from the preceding discussion and the fact that PosPol will position the polynomials if and only if \(p \in NC_{(3,11)}\). \(\square\)

### 4.4. Uniqueness results for one-letter algorithms.

In this section we investigate the possible variations in the matrices \(X\) and \(Y\) that are obtained by Alg.1 and Alg.2. We shall assume that the algorithms are applied to the homogeneous components of degree \(n\) in the family of polynomials \(\mathcal{P}\), where \(n \geq 2\) and is also the lowest degree of single letter monomials in \(\mathcal{P}\).

If \(\mathbb{P}\) denotes an array of the given \(k^2\) polynomials (as on the right hand side of (1.3)) that admits the nc representation
\[
\mathbb{P} = p(X, Y) = \sum_{\alpha, \beta} c_{\alpha, \beta} m_{\alpha, \beta}(X, Y), \quad \text{where } |\alpha| + |\beta| = n \text{ in the sum}
\]  
and \(\Pi\) is a \(k \times k\) permutation matrix, then
\[
\Pi^T \mathbb{P} \Pi = \sum_{\alpha, \beta} c_{\alpha, \beta} m_{\alpha, \beta}(\Pi^T X \Pi, \Pi^T Y \Pi).
\]

Moreover, since
\[
(X^{\alpha_1} Y^{\beta_1} \cdots X^{\alpha_r} Y^{\beta_r})^T = (Y^T)^{\beta_r} (X^T)^{\alpha_r} \cdots (Y^T)^{\beta_1} (X^T)^{\alpha_1},
\]
it is readily seen that
\[
\mathbb{P}^T = \sum_{\alpha, \beta} c_{\alpha, \beta} m_{\beta', \alpha'} (Y^T, X^T),
\]
where
\[ \beta = (\beta_1, \ldots, \beta_r) \implies \beta' = (\beta_r, \ldots, \beta_1) \quad \text{and} \quad \alpha = (\alpha_1, \ldots, \alpha_r) \implies \alpha' = (\alpha_r, \ldots, \alpha_1). \]

Below we shall show that, aside from a possible interchange of \(X\) and \(Y\), formulas (4.4) and (4.5) account for the only possible variation in the matrices \(X\) and \(Y\) that are generated by our algorithms. They correspond to the fact that the ParPosX algorithm allows for the diagonal variables to be placed in any order along the diagonal of \(X\) and the ambiguity in the next step of that algorithm, in which two variables \(x_u\) and \(x_v\) are arbitrarily assigned to be either the \(st\) entry or the \(ts\) entry of \(X\) (for some non ambiguous choice of \(s\) and \(t\) with \(s \neq t\)). This is in fact the only freedom that one has in positioning the variables within \(X\), i.e., once these \(k+2\) variables are allocated, the positions of the remaining variables in \(X\) are fully determined. This is substantiated by the next theorem.

Recall that the first matrix constructed by either Alg.1 or Alg.2 is always designated \(X\) (or \(\tilde{X}\)).

**Theorem 4.5.** Suppose that \(\mathcal{P}\) is a homogeneous family of polynomials \(p_1, \ldots, p_{k^2}\) of degree \(n\) in the commutative variables \(x_1, \ldots, x_{2k^2}\) that admits an nc representation. Then if \(X, Y\) and \(\tilde{X}, \tilde{Y}\) are generated by two different applications of Alg.1 or Alg.2, they are permutation equivalent (as defined in (1.13)).

**Proof.** Suppose first that the given family \(\mathcal{P}\) of \(k^2\) polynomials contains exactly \(k\) one letter monomials \(ax_{i_1}^{n_1}, \ldots, ax_{i_k}^{n_k}\) (all with the same coefficient \(a \in \mathbb{R} \setminus \{0\}\)) and let \(X\) and \(\tilde{X}\) denote the matrices that are determined by successive applications of either Alg.1 or Alg.2. Then the set of diagonal entries of \(X\) (without regard to to their positions on the diagonal) will coincide with the diagonal entries of \(\tilde{X}\). Moreover, since \(D2Pa1\) and \(D2Pa2\) depend only upon the diagonal variables and not upon how they are positioned along the diagonal, the set of \(k^2\) variables in \(X\) coincides with the set of \(k^2\) variables in \(\tilde{X}\).

Let
\[ L(x_{i_s}) = \{ x_u \in X \setminus \{ x_{i_s} \} : ax_{i_s}^{n_s-1}x_u \text{ appears in one of the polynomials in } \mathcal{P} \}. \]

There are \(2k-2\) distinct variables \(x_u\) in \(L(x_{i_s})\). Moreover, the selection of these terms is based totally on \(\mathcal{P}\) and not upon the position of \(x_{i_s}\) in \(X\). Therefore, if \(s \neq t\), then the two variables in the intersection
\[ L(x_{i_s}) \cap L(x_{i_t}) = \{ x_u, x_v \}, \]
are also independent of the position of \(x_{i_s}\) and \(x_{i_t}\).

If \(x_{i_s}\) is the \(ss\) entry of \(X\) and \(x_{i_t}\) is the \(tt\) entry of \(X\), then either \(x_u\) is the \(st\) entry and \(x_v\) is the \(ts\) entry, or vice versa. It is impossible to decide. However, once one of these two possibilities is chosen, then the position of all the remaining \(k^2-k-2\) variables in \(X\) are determined by the algorithm.

The algorithm ParPosX accomplishes this by inspecting terms of the form \(x_{i_j}^{n_j-2}x_tx_m\) in \(\mathcal{P}\), and so once one variable is placed, the algorithm is able to partition the set \(L(x_{i_s})\) into the sets
\[ R(x_{i_s}) = \{ x_u \in X : x_u \text{ is in the same row as } x_{i_s} \}. \]
and
\[ C(x_{is}) = \{x_u \in X : x_u \text{ is in the same column as } x_{is}\} \]
for each integer \( s, 1 \leq s \leq k \), solely by analyzing monomials contained in \( P \). Consequently, these sets do not depend on the position of the diagonal entry \( x_{is} \). Since \( \tilde{X} \) has the same diagonal elements as \( X \), there exists a permutation \( \pi \) of the integers \( \{1, \ldots, k\} \) such that if \( \tilde{x}_{ss} \) denotes the \( ss \) entry in the matrix \( \tilde{X} \) obtained in a second application of either of the two algorithms, then \( \tilde{x}_{ss} = x_{i\pi(s)} \) and correspondingly, if
\[
(4.6) \quad \Pi = \begin{bmatrix}
e_{\pi(1)}^T \\
\vdots \\
e_{\pi(k)}^T 
\end{bmatrix} = \sum_{j=1}^{k} e_j e_{\pi(j)}^T, \quad \text{where } e_i \text{ denotes the } i\text{-th column of } I_k,
\]
then clearly the diagonal entries of the matrices \( X \) and \( \Pi^T \tilde{X} \Pi \) will be positioned in the same way along the diagonal.

The fundamental observation is that
\[
(4.7) \quad \Pi^T E_{st} \Pi = \left( \sum_{i=1}^{k} e_{\pi(i)} e_i^T \right) e_s e_t^T \left( \sum_{j=1}^{k} e_j e_{\pi(j)}^T \right) = e_{\pi(s)} e_{\pi(t)}^T = E_{\pi(s), \pi(t)}.
\]

This accounts for the permutations. Transposition is a little more complicated. The point is that after the diagonal variables in \( X \) are positioned, say \( x_{is} \) is the \( ss \) entry of \( X \) for \( s = 1, \ldots, k \), as above, and if \( s \neq t \) and \( L(x_{is}) \cap L(x_{it}) = \{x_u, x_v\} \), then either \( x_u \) is \( st \) entry and \( x_v \) is the \( ts \) entry, or the other way around. Thus, if

1. If \( x_u \) is assigned to the \( st \) position of \( X \), then the polynomial containing terms of the form
   \[ ax_{is}^{n-1} x_u + ax_w x_{is}^{n-1} + ax_{is}^{n-2} x_w x_z + \cdots \]
   must be placed in the \( st \) position in the array \( \mathbb{P} \). But this means that either \( x_w \) is in the \( sr \) position or the \( rt \) position of \( X \) for some \( r \) other than \( s \) or \( t \), since the \( ss, tt \) and \( st \) positions are already occupied. (If it is in the \( rt \) position, then \( x_z \) will be in the \( sr \) position, so there is no loss of generality in assuming that \( x_w \) is in the \( sr \) position of \( X \).) Therefore, there is no loss of generality in assuming that \( x_w \in L(x_{is}) \cap L(x_{it}) \). Consequently,

   if \( x_u \) is put in the \( st \) position of \( X \), \( x_w \) will be in the \( sr \) position.

2. If \( x_u \) is placed in the \( ts \) position, then the polynomial displayed above must be placed in the \( ts \) position of the array \( \mathbb{P} \). Consequently \( x_w \) must be in either the \( tr \) position or the \( rs \) position. But since it is in \( L(x_{is}) \cap L(x_{it}) \), the only viable option is that it is in the \( rs \) position of \( X \):

   if \( x_u \) is put in the \( ts \) position of \( X \), \( x_w \) will be in the \( rs \) position.

Thus, transposition of the two variables in the first step after the diagonals are fixed, moves \( X \) to \( X^T \).

Now suppose that there are two sets of single letter monomials \( ax_{i_1}^{n_1}, \ldots, ax_{i_k}^{n_k} \) and \( bx_{j_1}^{n_1}, \ldots, bx_{j_k}^{n_k} \), where the monomials \( ax_{i_k} \) and \( bx_{j_k} \) must occur in the same polynomial
in \( p \). In this case it is possible for \( X \) to have diagonal variables \( x_{i_1}, \cdots, x_{i_k} \) and for \( \widetilde{X} \) to have diagonal variables \( x_{j_1}, \cdots, x_{j_k} \). Therefore, we must show that if this happens that \( X \) is pt equivalent to \( \widetilde{Y} \) and that \( \widetilde{X} \) is pt equivalent to \( Y \).

If \( x_{i_1}, \cdots, x_{i_k} \) are the diagonal variables for \( X \), then \( x_{j_1}, \cdots, x_{j_k} \) must be the diagonal variables of \( Y \) and \( x_{i_s} \) and \( x_{j_s} \) must occur in the same diagonal position of \( X \) and \( Y \). Similarly, if \( x_{j_1}, \cdots, x_{j_k} \) are the diagonal variables of \( \widetilde{X} \) and \( x_{i_1}, \cdots, x_{i_k} \) are the diagonal variables of \( \widetilde{Y} \), then \( x_{j_s} \) and \( x_{i_s} \) must occur in the same diagonal position in \( \widetilde{X} \) and \( \widetilde{Y} \). As in (4.7), let \( \Pi \) be the permutation matrix such that the diagonal entries of \( \Pi^T X \Pi \) and \( \widetilde{Y} \) are positioned in the same way along the diagonal. It follows that the diagonal entries of \( \Pi^T Y \Pi \) and \( \widetilde{X} \) are positioned in the same way as well.

When Alg.1 or Alg.2 determines the sets \( L(x_{i_s}) \) for \( X \) and \( L(x_{j_s}) \) for \( \widetilde{X} \) for \( 1 \leq s \leq k \), it does so by considering terms of the form \( ax_{i_s}^{n-1}x_u \) and \( bx_{j_s}^{n-1}x_v \). One readily sees that for each term of the form \( ax_{i_s}^{n-1}x_u \) that occurs in a polynomial in \( \mathcal{P} \), there is a corresponding term \( bx_{j_s}^{n-1}x_v \) that occurs in the same polynomial. Therefore the fundamental observation is that for each term \( x_u \in L(x_{i_s}) \), there is a corresponding term \( x_v \in L(x_{j_s}) \) and furthermore, if \( x_u \) is placed in the \( st \) position in \( X \), Algorithm PolyPos will place \( x_v \) in the \( st \) position in \( Y \). Similarly, if \( x_v \) is placed in the \( st \) position in \( \widetilde{X} \), this correspondence ensures that Algorithm PolyPos will place \( x_u \) in the \( st \) position in \( \widetilde{Y} \). Thus the nc variables \( X \) and \( Y \) are pt equivalent to \( \widetilde{X} \) and \( \widetilde{Y} \).

Now that we have shown that there is a strong relationship between any two pairs of matrices constructed by our algorithms, we want to exploit this relationship to construct nc polynomials for matrix pairs determined by our algorithms. In particular, if \( X, Y \) and \( \widetilde{X}, \widetilde{Y} \) are two pairs of matrices determined by our algorithms for a given family \( \mathcal{P} \) and \( p(X, Y) \) is an nc representation of \( \mathcal{P} \), we would like to conclude that there exists an nc polynomial \( \tilde{p} \) such that \( \tilde{p}(\widetilde{X}, \widetilde{Y}) \) is also an nc representation of \( \mathcal{P} \). We shall see in Lemma 4.6 below that this is true.

Lemma 4.6 supplements Theorem 1.5 and is formulated in terms of a pair of auxiliary nc polynomials that are expressed in terms of the notation introduced in (4.5):

\[
(4.8) \quad p_t(X, Y) = \sum_{\alpha, \beta} c_{\alpha, \beta} m_{\beta, \alpha'}(Y, X) \quad \text{and} \quad p(X, Y) = p(Y, X).
\]

Then it is clear that

\[
p_t(X^T, Y^T) = p(X, Y)^T.
\]

Lemma 4.6. Suppose that \( p_1, \cdots, p_{k_2} \) is a collection of polynomials \( \mathcal{P} \) with an nc representation \( p(X, Y) \) satisfying the conditions in Theorem 1.4. If DiagPar1 or DiagPar2, ParPosX and PosY generate a pair of matrices \( \widetilde{X} \) and \( \widetilde{Y} \), then there exists a nc polynomial \( \tilde{p} \) such that \( \tilde{p}(\widetilde{X}, \widetilde{Y}) \) is an nc representation of \( \mathcal{P} \).

Proof. If \( p \) satisfies the conditions in Theorem 1.4, then the algorithms DiagPar1 or DiagPar2, ParPosX, PolyPos and PosY can be applied to construct \( X \) and \( Y \). Since
\( p(X, Y) \) is an nc representation of \( \mathcal{P} \), by Theorem 4.5 and equations (4.3), (4.5), and (4.8) the following must hold:

\[
\begin{align*}
(1) & \quad X = \Pi^T \tilde{X} \Pi, \quad Y = \Pi^T \tilde{Y} \Pi, \quad \Rightarrow p(\tilde{X}, \tilde{Y}) \text{ is an nc rep.} \\
(2) & \quad X = \Pi^T \tilde{X}^T \Pi, \quad Y = \Pi^T \tilde{Y}^T \Pi, \quad \Rightarrow p_t(\tilde{X}, \tilde{Y}) \text{ is an nc rep.} \\
(3) & \quad X = \Pi^T \tilde{Y} \Pi, \quad Y = \Pi^T \tilde{X} \Pi, \quad \Rightarrow p(\tilde{X}, \tilde{Y}) \text{ is an nc rep.} \\
(4) & \quad X = \Pi^T \tilde{Y}^T \Pi, \quad Y = \Pi^T \tilde{X}^T \Pi \quad \Rightarrow p_t(\tilde{X}, \tilde{Y}) \text{ is an nc rep.}
\end{align*}
\]

(4.9)

Using the uniqueness results developed in this section, we can now prove Theorem 1.5.

4.4.1. Proof of Theorem 1.5.

Proof. Given that \( p(X, Y) \) and \( \tilde{p}(\tilde{X}, \tilde{Y}) \) satisfy the conditions in Theorem 1.4, Algorithms ParPos1 or ParPos2, ParPosX, PolyPos, and PosY can successfully determine the pairs \( X, Y \) and \( \tilde{X}, \tilde{Y} \). Therefore, we may apply Theorem 4.5 to obtain the desired result.

□

4.5. Determining \( p(X, Y) \) given \( X, Y \) and the positions of the polynomials in \( \mathcal{P} \).

Once the matrices \( X \) and \( Y \) and the positions of the polynomials in \( \mathcal{P} \) are determined by the previous algorithms, it remains only to find an nc representation for \( \mathcal{P} \). This rests on the following elementary observation, which is formulated in terms of the notation introduced in (2.3):

Lemma 4.7. The monomial \( m_{\alpha,\beta}(X, Y) \) is a \( k \times k \) array of polynomials in the variables \( x_1, \ldots, x_{2k^2} \) of degree \(|\alpha| + |\beta|\).

Proof. This is immediate from the rules of matrix multiplication.

□

4.5.1. Algorithm NcCoef. Suppose that

\[
\mathcal{P} = \{p_1, \cdots, p_{k^2}\} \quad \text{is a family of } k^2 \text{ polynomials}
\]

of degree \( d \) or less in the commutative variables \( x_1, \ldots, x_{2k^2} \) and that these variables are positioned in \( X \) and \( Y \) and the polynomials in \( \mathcal{P} \) are positioned in an array

\[
\mathbb{P} = \begin{bmatrix}
  p_{\lambda(1)} & \cdots & p_{\lambda(k)} \\
  \vdots & \ddots & \vdots \\
  p_{\lambda(k^2-k+1)} & \cdots & p_{\lambda(k^2)}
\end{bmatrix}
\]

as in (1.3). Then, since any nc polynomial \( p(X, Y) \) in the variables \( X \) and \( Y \) of degree \( d \) can be expressed in terms of monomials \( m_{\alpha,\beta}(X, Y) \) as

\[
p(X, Y) = \sum_{|\alpha|+|\beta| \leq d} c_{\alpha,\beta} m_{\alpha,\beta}(X, Y),
\]

the NcCoef algorithm reduces to solving the system of linear equations

\[
p(X, Y) = \mathbb{P},
\]
for the unknown coefficients \(c_{\alpha,\beta}\). This system of equations has a solution if and only if there exists an nc polynomial \(p\) such that \(p(X,Y)\) is an nc representation of \(P\). In view of Lemma 4.7, the addition of monomials \(m_{\alpha,\beta}(X,Y)\) of degree higher than \(d\) in (4.12) does not effect the solvability of (4.13).

**Remark 4.8.** Algorithm NcCoef determines whether or not an nc representation exists for a given pair of nc variables \(X\) and \(Y\). Klep and Vinnikov [10] have been working on various elegant abstract characterizations of those sets of \(X, Y, \) and \(P\) which admit an nc representation. However, their work [10] does not address the issue of implementing tests for these characterizations.

The following example illustrates how Algorithm NcCoef can be applied to determine \(p(X,Y)\) for a given family \(P\).

**Example 4.9.** Let
\[
p_1 = 4x_1^2 + 4x_2x_3 + 2x_1x_5 + 6x_5^2 + x_3x_6 + 6x_6x_7
\]
\[
p_2 = 4x_1x_2 + 4x_2x_4 + x_2x_5 + x_1x_6 + 4x_4x_6 + 6x_5x_6 + x_2x_8 + 6x_6x_8
\]
\[
p_3 = 4x_1x_3 + 4x_3x_4 + x_3x_5 + x_1x_7 + 4x_4x_7 + 6x_5x_7 + x_3x_8 + 6x_7x_8
\]
\[
p_4 = 4x_2x_3 + 4x_4^2 + x_3x_6 + x_2x_7 + 6x_6x_7 + 2x_4x_8 + 6x_2^2.
\]

and suppose that
\[
(4.14) \quad X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}, \quad Y = \begin{pmatrix} x_5 & x_6 \\ x_7 & x_8 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} p_1 & p_2 \\ p_3 & p_4 \end{pmatrix}.
\]

Our goal is to find an nc representation \(p\) or to refute its existence.

**Discussion** Since the given family of polynomials is homogeneous of degree two, the first step of Algorithm NcCoef is to form the polynomial
\[
aX^2 + bXY + cYX + dY^2 = p(X,Y),
\]
and then set \(p(X,Y) = P\). This yields four relations, one for each entry:
\[
ax_1^2 + ax_2x_3 + bx_1x_5 + cx_1x_5 + dx_5^2 + cx_3x_6 + dx_6x_7 = p_1
\]
\[
ax_1x_2 + ax_2x_4 + cx_2x_5 + bx_1x_6 + cx_4x_6 + dx_5x_6 + bx_2x_8 + dx_6x_8 = p_2
\]
\[
ax_1x_3 + ax_3x_4 + bx_3x_5 + cx_1x_7 + bx_4x_7 + dx_5x_7 + cx_3x_8 + dx_7x_8 = p_3
\]
\[
ax_2x_3 + ax_4^2 + bx_3x_6 + cx_2x_7 + dx_6x_7 + bx_4x_8 + cx_4x_8 + dx_8^2 = p_4.
\]

Upon matching the coefficients of the left hand side of the first row in the preceding array with those of \(p_1\), we readily obtain the list of linear equations
\[
a = 4, \quad b + c = 2, \quad c = 1 \quad \text{and} \quad d = 6,
\]
the unknowns must satisfy; hence \(a = 4, b = 1, c = 1, d = 6\). It is then easily checked that this choice of coefficients works for the remaining three rows of the array. Therefore, for this particular choice of \(X\) and \(Y\) and positioning of the polynomials \(p_1, p_2, p_3, p_4\),
\[
p(X,Y) = 4X^2 + XY + YX + 6Y^2.
\]

□
4.5.2. Homogeneous sorting and Implementation of NcCoef. In order to implement the NcCoef algorithm, it is convenient to first sort each polynomial $p_j$ in $\mathcal{P}$ as a sum

$$p_j = \sum_{i=1}^{d} (p_j)_{[i]}(x_1, \ldots, x_{2k^2})$$

of homogeneous polynomials $(p_j)_{[i]}$ of degree $i$ in which $(p_j)_{[i]}$ is the sum of terms in $p_j$ that are homogeneous of degree $i$ and is taken equal to zero if there are no such terms.

Let

$$\mathcal{P}_i = \{(p_1)_{[i]}, \ldots, (p_{k^2})_{[i]}\}$$

and, if at least one of the polynomials in $\mathcal{P}_i$ is nonzero, try to find a homogeneous nc polynomial representation $p_i(X,Y)$ of degree $i$ for $\mathcal{P}_i$. We shall refer to this procedure as homogeneous sorting.

An obvious consequence of Lemma 4.7 is:

**Lemma 4.10.** A family $\mathcal{P}$ of $k^2$ polynomials of degree $\leq d$ in $2k^2$ commuting variables admits an nc representation $p(X,Y)$ if and only if $\mathcal{P}_i$ admits an nc representation $p_i(X,Y)$ with the same $X$ and $Y$ for each $1 \leq i \leq d$.

The primary advantage of homogeneous sorting is apparent when implementing Algorithms DiagPar1, DiagPar2, ParPosX, PolyPos, PosY and NcCoef. One applies these algorithms to the nonzero family $\mathcal{P}_i$ for each $i$ separately. Typically, to save computational cost, choose $i$ as small as possible in order to minimize computations. Then, once $X$, $Y$ and the position of the polynomials are determined, it is easy to fill in the coefficients of the terms in $p_i(X,Y)$ by comparison with the array of terms $\{(p_1)_{[i]}, \ldots, (p_{k^2})_{[i]}\}$ for the remaining choices of $i$, one degree at a time, just as in Example 4.9. It is important that the same $X$ and $Y$ are used for each choice of $i$. Now we give a cautionary example.

**Example 4.11.** If $p_1, \ldots, p_4$ are as in Example 4.9, then the set of polynomials

$$q_1 = p_1 + (x_2^2 + x_2x_3)x_1 + (x_1x_2 + x_2x_5)x_3$$

$$q_2 = p_2 + (x_2^2 + x_2x_3)x_2 + (x_1x_2 + x_2x_5)x_5$$

$$q_3 = p_3 + (x_3x_1 + x_5x_3)x_1 + (x_3x_2 + x_5^2)x_3$$

$$q_4 = p_4 + (x_3x_1 + x_5x_3)x_2 + (x_3x_2 + x_5^2)x_5$$

does not admit an nc representation even though the terms of degree two admit an nc representation and the terms of degree three admit an nc representation.

**Discussion** The terms of degree two are exactly the polynomials $p_1, \ldots, p_4$ considered in in Example 4.9 and hence either lead to the representation considered there, or to an equivalent representation that corresponds to conjugation of the matrices $X$, $Y$ and the polynomial array matrix $\mathbb{P}$ by a $2 \times 2$ permutation matrix $\Pi$ (to obtain $\Pi^T X \Pi$, $\Pi^T Y \Pi$ and $\Pi^T \mathbb{P} \Pi$ in place of $X$, $Y$ and $\mathbb{P}$), or transposition or to an interchange of $X$ and $Y$. But in all these shufflings, the variables $\{x_1, x_2, x_3, x_4\}$ will belong to one of the matrices and the remaining variables $\{x_5, x_6, x_7, x_8\}$ will belong to the other.
The polynomials \( q_1, \ldots, q_4 \), will not admit an nc representation because the added terms come from
\[
\begin{bmatrix}
  x_1 & x_2 \\
  x_3 & x_5
\end{bmatrix}^3
\]
which involves a mixing of the variables in the matrices \( X \) and \( Y \) that are obtained by analyzing \( p_1, \ldots, p_4 \).

4.5.3. Effectiveness of NcCoef. The final task in our analysis of Algorithm NcCoef is to show that it will either be successful for all pairs \( X, Y \) produced our algorithms or it will fail for all such pairs. The next proposition (4.12) characterizes the nc polynomials \( p \) such that \( p(X, Y) \) is an nc representation of \( P \) when \( X \) and \( Y \) are determined by Algorithms DiagPar1 or DiagPar2, ParPosX PolyPos and PosY. Proposition 4.13 then insures that a given family \( P \) will either have an nc representation \( p(X, Y) \) for all \( X, Y \) produced by our algorithms or will have no representation for any such \( X \) and \( Y \).

**Proposition 4.12.** Suppose that a given family \( P \) of \( k^2 \) polynomials in \( 2k^2 \) commuting variables admits an nc representation \( p(X, Y) \) and that the algorithms DiagPar1 or DiagPar2, ParPosX PolyPos and PosY produce the matrices \( X \) and \( Y \). Then \( p \) belongs to the set \( W \) of nc polynomials that is specified in Theorem 1.4.

**Proof.** The coefficient conditions defining \( W \) are the exact conditions required for the algorithms DiagPar1 or DiagPar2, ParPosX, PolyPos and PosY to be successful. Furthermore, these algorithms will only be successful if these conditions are satisfied. Therefore, the assumption that our algorithms generate \( X \) and \( Y \) implies that if \( p \) exists, then \( p \in W \).

**Proposition 4.13.** Suppose that \( P \) is a family of \( k^2 \) polynomials in \( 2k^2 \) commuting variables that satisfies (4.10). Let \( X, Y \) and \( \tilde{X}, \tilde{Y} \) be distinct pairs of matrices determined by separate applications of DiagPar1 or DiagPar2, ParPosX, PolyPos and PosY. Then Algorithm NcCoef will successfully determine an nc representation \( p(X, Y) \) of \( P \) if and only if it successfully determines an nc representation \( \tilde{p}(\tilde{X}, \tilde{Y}) \) of \( P \).

**Proof.** If the NcCoef algorithm produces an nc representation \( p(X, Y) \) of \( P \), then Proposition 4.12 implies that \( p \) belongs to the set \( W \) specified in Theorem 1.4. Then Lemma 4.6 implies that there exists an nc polynomial \( \tilde{p} \in W \) such that \( \tilde{p}(\tilde{X}, \tilde{Y}) \) is an nc representation of \( P \). The reverse direction follows by a similar argument.

4.6. The Size and Cost of NcCoef. We now determine the cost required to implement NcCoef. For now, we neglect the cost to form the linear systems associated with NcCoef and only focus on the cost of solving these systems.

For given arrangements \( \sigma \) and \( \lambda \) of the variables \( x_1, \ldots, x_{2k^2} \) in \( X \) and \( Y \) and the polynomials \( p_1, \ldots, p_{k^2} \) in \( P \) we apply algorithm NcCoef to obtain a system of equations in the undetermined variables \( c_{\alpha, \beta} \) as in (4.12) and (4.13). By applying the method of homogeneous sorting as described in Section 4.5.2, we obtain \( (d - 1) \) systems of equations \( M_i \) in the unknowns \( c_{\alpha, \beta} \) formed by equating
\[
\mathbb{P}_\lambda^i = p_\alpha^i(X, Y),
\]
where \( \underline{P}_{i}^{i} \) and \( p_{\sigma}^{i}(X, Y) \) are formed from (4.11) and (4.12).

To determine the cost of solving this linear system let

\[
(4.21) \quad t_{ij} = \text{number of monomials in } p_{\lambda(ij)}^{i},
\]

and set \( \tau_{i} = \sum_{j=1}^{k_{i}} t_{ij} \). The number of noncommutative monomials in the variables \( X \) and \( Y \) of degree \( i \) is \( 2^{i} \). Therefore the number of unknowns \( c_{\alpha\beta} \) in our system corresponding to homogeneous terms of degree \( i \) will be \( 2^{i} \). It follows that

\[
M_{i} \quad \text{is a } \tau_{i} \times 2^{i} \quad \text{system of equations}
\]

in the unknowns \( c_{\alpha\beta} \) satisfying \(|\alpha + \beta| = i\).

When \( \tau_{i} > 2^{i} \), \( M_{i} \) will be overdetermined. The cost of solving this system using an LU decomposition is

\[
(4.22) \quad k^{2} \tau_{i} 4^{i} - \frac{8^{i}}{3} \quad \text{arithmetic operations (see [7] §3.2.11)}
\]

We must solve such a system for each \( 2 \leq i \leq d \), so the total cost to solve the linear system when each \( k^{2} \tau_{i} > 2^{i} \) is

\[
(4.23) \quad \text{TotLinC}_{\sigma\lambda} \leq \sum_{i=2}^{d} \left( \tau_{i} 4^{i} - \frac{8^{i}}{3} \right) \quad \text{arithmetic operations.}
\]

Similarly, when \( \tau_{i} \leq 2^{i} \) we get

\[
(4.24) \quad \text{TotLinC}_{\sigma\lambda} \leq \sum_{i=2}^{d} \frac{2^{3i+1}}{3} \quad \text{arithmetic operations (see [7] §3.2.9)}.
\]

4.7. Final Results for One letter Algorithms. To this point we have developed the following algorithms:

**DiagPar1:** Partitions the commutative variables between the matrices \( X \) and \( Y \) and works under the assumption that \( \varphi(n, 0) \neq 0 \) for some \( n \geq 2 \) and \( n \varphi(n, 0) \neq \varphi(n - 1, 1) \).

**DiagPar2:** Partitions the commutative variables between the matrices \( X \) and \( Y \) and works under the assumption that \( \varphi(n, 0) \neq 0 \) for some \( n \geq 2 \) and \( \varphi(n, 0) \neq \varphi(n - 1, 1; Y) \) or \( \varphi(n, 0) \neq \varphi(Y; n - 1, 1) \).

**ParPosX:** Positions the variables in the matrix \( X \) if \( \varphi(n, 0) \neq 0 \) for some \( n \geq 2 \) and the commutative variables are partitioned.

**PosPol:** Positions the polynomials in the family if \( \varphi(n, 0) \neq 0 \) for some \( n \geq 2 \) and the commutative variables are partitioned and positioned in \( X \).

**PosY** Positions the commutative variables in \( Y \) if \( \varphi(0, n) \neq 0 \) for some \( n \geq 1 \), the variables are partitioned and positioned in \( X \) and the polynomials are positioned.

**NcCoef** Given \( X, Y \) and the positioning of the \( p_{j} \) in a matrix, this algorithm determines whether or not an nc \( p \) exists such that \( p(X, Y) \) generates the matrix containing the \( p_{j} \).
Remark 4.16. There are analogues of Theorems 4.14 and 4.15 in which $P_{nc}$ representation of algorithm $NcCoef$ will be successful. Therefore, Alg.2 will successfully determine an nc representation of $X_{NC}$.

Theorem 4.14. Let $P$ be a family of $k^2$ polynomials in $2k^2$ commuting variables. Then Alg.1 yields an nc representation $p(X, Y)$ of $P$ if and only if the given family $P$ admits an nc representation in $NC_{d}$.

Proof. If Alg.1 determines an nc representation $p(X, Y)$, then Proposition 4.12 implies that $p \in NC_{d}$. Now suppose that $P$ admits an nc representation in $NC_{d}$. Then Theorem 3.1, Theorem 4.1, Proposition 4.3, and Proposition 4.4 imply that Alg.1 will successfully determine a pair of nc variables $X$ and $Y$. Moreover, Lemma 4.6 implies that there exists an nc polynomial $p$ such that $p(X, Y)$ is an nc representation of $P$. But this implies that for this choice of $X$ and $Y$ that Algorithm $NcCoef$ will be successful. Furthermore, Proposition 4.13 implies that for any pair of nc variables produced by Alg.1, algorithm $NcCoef$ will be successful. Therefore, Alg.1 will successfully determine an nc representation of $P$.

Let $NC_{d}$ denote the class of all nc polynomials $p(X, Y)$ of degree $d > 1$ for which there exists integers $s \geq 0$, $t \geq 1$ and $n \geq 2$ such that

\[
\begin{align*}
\varphi(s, t) &\neq 0, \\
\varphi(n, 0) &\neq 0, \\
\varphi(n, 0) &\neq \varphi(0, n),
\end{align*}
\]

and

\[
\begin{align*}
n\varphi(n, 0) &\neq \varphi(n - 1, 1).
\end{align*}
\]

Theorem 4.15. Let $P$ be a family polynomials in $2k^2$ commuting variables. Then Alg.2 yields an nc representation $p(X, Y)$ of $P$ if and only if the given family $P$ admits an nc representation in $NC_{d}$.

Proof. If Alg.2 determines an nc representation $p(X, Y)$, then Proposition 4.12 implies that $p \in NC_{d}$. Now suppose that $P$ admits an nc representation in $NC_{d}$. Then Theorem 3.2, Theorem 4.1, Proposition 4.3, and Proposition 4.4 imply that Alg.2 will successfully determine $X$ and $Y$. Furthermore, Lemma 4.6 implies that there exists an nc polynomial $p$ such that $p(X, Y)$ is an nc representation of $P$. But this implies that for this choice of $X$ and $Y$ that Algorithm $NcCoef$ will be successful. Furthermore, Proposition 4.13 implies that for any pair of nc variables produced by Alg.2, algorithm $NcCoef$ will be successful. Therefore, Alg.2 will successfully determine an nc representation of $P$.

Remark 4.16. There are analogues of Theorems 4.14 and 4.15 in which $NC_{d}$ is replaced by the class $NC_{d}$ of all nc polynomials of degree $d > 1$ for which there exists integers $s \geq 1$, $t \geq 0$ and $n \geq 2$ such that

\[
\begin{align*}
\varphi(s, t) &\neq 0, \\
\varphi(0, n) &\neq 0, \\
\varphi(n, 0) &\neq \varphi(0, n)
\end{align*}
\]

and

\[
\begin{align*}
n\varphi(0, n) &\neq \varphi(1, n - 1).
\end{align*}
\]
and NC(4.26) is replaced by the class NC\(_{(4.28)}\) of all nc polynomials of degree \(d > 1\) for which there exists integers \(s \geq 1\), \(t \geq 0\) and \(n \geq 2\) such that

\[
\begin{align*}
\varphi(s, t) &\neq 0, \quad \varphi(0, n) \neq 0, \quad \varphi(n, 0) \neq \varphi(0, n), \\
\varphi(0, n) &\neq \varphi(1, n - 1; Y), \quad \text{or} \quad \varphi(0, n) \neq \varphi(Y; 1, n - 1).
\end{align*}
\]

5. Examples

This section is devoted to a number of examples to illustrate the algorithms that were developed in earlier sections, as well as some variations thereof.

**Example 5.1.** The set of 4 polynomials

\[
\begin{align*}
p_1 &= x_1x_5 + 5x_5^2 + x_2x_7 + 5x_6x_7 \\
p_2 &= x_1x_6 + 5x_5x_6 + x_2x_8 + 5x_6x_8 \\
p_3 &= x_3x_5 + x_4x_7 + 5x_5x_7 + 5x_7x_8 \\
p_4 &= x_3x_6 + 5x_6x_7 + x_4x_8 + 5x_8^2
\end{align*}
\]

in the commutative variables \(x_1, \ldots, x_8\) can be identified with the entries in an nc polynomial \(p(X,Y)\) for appropriate choices of the \(2 \times 2\) matrices \(X\) and \(Y\). The objective is to find such an identification.

**Discussion** Since these 4 polynomials are homogeneous of degree 2, it is reasonable to look for an nc representation must be of the form

\[p(X,Y) = aX^2 + bXY + cYX + dY^2\]

for some choice of \(a, b, c, d \in \mathbb{R}\). Moreover, since there are only two one letter monomials in the given family of polynomials:

- \(5x_5^2\) in \(p_1\) and \(5x_8^2\) in \(p_4\),

the corresponding variables must sit on the diagonal of either \(X\) or \(Y\). We shall arbitrarily place \(x_8\) in the 11 position of \(X\) and \(x_5\) in the 22 position of \(X\). This in turn forces \(p_1\) to be in the 22 position of \(p(X,Y)\) and \(p_4\) to be in the 11 position of \(p(X,Y)\) and forces \(a = 5\) and \(d = 0\). Furthermore, if \(A = X\) and \(B = Y\) with \(x_5 = x_8 = \alpha\), one of the other variables \(x_i = \beta\) and the remaining five variables equal to zero, then \(AB = BA\) and

\[p(A, B) = 5A^2 + (b + c)AB,\]

i.e., in terms of the notation introduced in subsection 1.3.2 with \(c_{ij} = \varphi(i, j)\) for short, \(c_{20} = 5, c_{11} = b + c\) and \(c_{02} = 0\). Consequently, there are six possibilities:

1) If \(x_i\) is the 12 entry of \(X\), then

\[
A = \begin{bmatrix} \alpha & \beta \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 & 10\alpha\beta \\ 0 & 5\alpha^2 \end{bmatrix}.
\]

2) If \(x_i\) is the 12 entry of \(Y\), then

\[
A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 & \beta \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 & 0 \\ 0 & c_{11}\alpha\beta \end{bmatrix}.
\]
3) If \( x_i \) is the 21 entry of \( X \), then
\[
A = \begin{bmatrix} \alpha & 0 \\ \beta & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 & 0 \\ 10\alpha \beta & 5\alpha^2 \end{bmatrix}.
\]

4) If \( x_i \) is the 21 entry of \( Y \), then
\[
A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & \beta \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 & 0 \\ c_{11} \alpha \beta & 5\alpha^2 \end{bmatrix}.
\]

5) If \( x_i \) is the 11 entry of \( Y \), then
\[
A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} \beta & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 + c_{11} \alpha \beta & 0 \\ 0 & 5\alpha^2 \end{bmatrix}.
\]

6) If \( x_i \) is the 22 entry of \( Y \), then
\[
A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & \beta \end{bmatrix} \quad \text{and} \quad p(A, B) = \begin{bmatrix} 5\alpha^2 & 0 \\ 0 & 5\alpha^2 + c_{11} \alpha \beta \end{bmatrix}.
\]

Thus, if:

1. \( x_i \) is an off-diagonal entry of \( X \), 2 polynomials will equal \( 5\alpha^2 \), one will equal \( 10\alpha \beta \) and one will equal 0.
2. \( x_i \) is an off-diagonal entry of \( Y \), 2 polynomials will equal \( 5\alpha^2 \), one will equal \( c_{11} \alpha \beta \) and one will equal 0.
3. \( x_i \) is a diagonal entry of \( Y \), one polynomial will equal \( 5\alpha^2 \), one will equal \( 5\alpha^2 + c_{11} \alpha \beta \) and two will equal 0.

Correspondingly, if \( x_5 = x_8 = \alpha \) and \( x_i = \beta \) and all the other variables are set equal to zero, then the polynomials \( p_1, \ldots, p_4 \) assume the values shown in the following array
\[
\begin{bmatrix}
   i = 1 & i = 2 & i = 3 & i = 4 & i = 6 & i = 7 \\
   p_1 = & 5\alpha^2 + \alpha \beta & 5\alpha^2 & 5\alpha^2 & 5\alpha^2 & 5\alpha^2 \\
   p_2 = & 0 & \alpha \beta & 0 & 0 & 10\alpha \beta \\
   p_3 = & 0 & 0 & \alpha \beta & 0 & 0 & 10\alpha \beta \\
   p_4 = & 5\alpha^2 & 5\alpha^2 & 5\alpha^2 & 5\alpha^2 + \alpha \beta & 5\alpha^2 & 5\alpha^2 
\end{bmatrix}.
\]

Upon comparing the values of the polynomials \( p_1, \ldots, p_4 \) for the 6 possible choices of \( x_i = \beta \) with the possibilities (1)–(3) indicated just above, it is readily seen (from setting (3)) that \( x_1 \) and \( x_4 \) are diagonal entries in \( Y \) and \( c_{11} = 1 \). (In fact since \( p_1 \) is in the 22 position of \( p(X, Y) \), \( x_1 \) must be in the 22 position of \( Y \), which forces \( x_4 \) to be in the 11 position of \( Y \).) The variables \( x_2 \) and \( x_3 \) will be off-diagonal entries of \( Y \) by setting (2); and hence the remaining variables \( x_6 \) and \( x_7 \) must be off-diagonal entries in \( X \). A more detailed analysis would serve to position these last 4 variables in \( X \) and \( Y \), after fixing one of them; see Remark 5.2.

**Remark 5.2.** As \( p_1 = x_1 x_5 + \cdots \), \( p_1 \) is in the 22 position of \( p(X, Y) \) and \( x_5 \) is in the 22 position of \( X \), it follows that \( x_1 \) is in the 22 position of \( Y \). Similarly, since \( p_4 = x_4 x_8 + \cdots \), \( p_4 \) is in the 11 position of \( p(X, Y) \) and \( x_8 \) is in the 11 position of \( X \), it follows that \( x_4 \) is in the 11 position of \( Y \). Moreover, since the term \( 5x_6 x_7 \) in \( p_1 \) (and \( p_4 \)) can only come from \( 5X^2 \), it follows that \( x_6 \) and \( x_7 \) must belong to \( X \). Consequently, the remaining two variables, \( x_2 \) and \( x_3 \), must belong to \( Y \).
To go further, assume that \( x_6 \) is in the 12 position of \( X \). Then \( x_7 \) must be in the 21 position of \( X \) and the polynomial
\[
p_2 = x_1 x_6 + \cdots \text{ is in the 12 position of } p(X,Y).
\]
Therefore, \( x_2 \) is in the 12 position of \( Y \) and the remaining variable \( x_4 \) is in the 21 position of \( Y \).

To better illustrate the algorithms, we shall return to the case where it is only known that \( x_j \in Y \) for \( j = 1, \ldots, 4 \), \( x_j \in X \) for \( j = 5, \ldots, 8 \), \( x_8 \) is the 11 position of \( X \) and \( x_5 \) is in the 22 position of \( X \). Then \( L_1 = \{ x_6, x_7 \} = L_2 \) and the positions of these two variables in \( X \) are not uniquely determined. We shall arbitrarily place \( x_6 \) in the 12 position of \( X \). Then \( x_7 \) must be in the 21 position and
\[
5X^2 = 5 \begin{bmatrix} x_8 & x_6 \\ x_7 & x_5 \end{bmatrix}^2 = 5 \begin{bmatrix} x_8^2 + x_6 x_7 & x_8 x_6 + x_6 x_5 \\ x_7 x_8 + x_5 x_7 & x_5^2 + x_7 x_6 \end{bmatrix}.
\]
Thus, \( p_2 = 5x_8 x_6 + 5x_6 x_5 + \cdots \) must sit in the 12 position of \( p(X,Y) \). Therefore, \( x_2 x_8 \) is also in the 12 position of \( p(X,Y) \), which forces \( x_2 \) to be in the 12 position of \( Y \). Therefore,
\[
X = \begin{bmatrix} x_8 & x_6 \\ x_7 & x_5 \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} x_4 & x_2 \\ x_3 & x_1 \end{bmatrix}.
\]
It is now readily checked that
\[
\begin{bmatrix} p_4 & p_2 \\ p_3 & p_1 \end{bmatrix} = 5X^2 + XY.
\]

Note!

\[
\begin{bmatrix} p_4 & p_2 \\ p_3 & p_1 \end{bmatrix} \neq 5X^2 + YX.
\]

Example 5.3. The polynomials
\[
p_1 = x_1^2 + x_2 x_3 - x_3 x_6 + x_2 x_7 \\
p_2 = x_1 x_2 + x_2 x_4 - x_2 x_5 + x_1 x_6 - x_4 x_6 + x_2 x_8 \\
p_3 = x_1 x_3 + x_3 x_4 + x_3 x_5 - x_1 x_7 + x_4 x_7 - x_3 x_8 \\
p_4 = x_2 x_3 + x_4^2 + x_3 x_6 - x_2 x_7
\]

admit an nc representation.

Discussion Since these 4 polynomials are homogeneous of degree 2, it is reasonable to look for an nc representation must be of the form
\[
p(X, Y) = aX^2 + bXY + cYX + dY^2
\]
for some choice of \( a, b, c, d \in \mathbb{R} \), just as in Example 5.1. Moreover, since there are only two one letter monomials
\[
x_1^2 \text{ in } p_1 \quad \text{and} \quad x_4^2 \text{ in } p_4,
\]
the corresponding variables are placed on the diagonal of $X$. Thus, $a = 1$ and $d = 0$, and the substitutions $X = A$ and $Y = B$ in the candidate $p(X, Y)$ for the nc representation yields the formula

$$p(A, B) = c_{20}A^2 + c_{11}AB + c_{02}B^2,$$

with $c_{20} = 1$, $c_{11} = b + c$ and $c_{02} = 0$.

We shall arbitrarily place $x_4$ in the 11 position of $X$ and $x_1$ in the 22 position of $X$. This forces $p_4$ and $p_1$ to be in the 11 and 22 positions of $p(X, Y)$, respectively. Then, setting $x_1 = x_4 = \alpha$ and one of the other variables $x_i = \beta$ leads to the following sets of values:

$$\begin{bmatrix}
  x_2 = \beta & x_3 = \beta & x_5 = \beta & x_6 = \beta & x_7 = \beta & x_8 = \beta \\
  p_1 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 \\
  p_2 & 2\alpha\beta & 0 & 0 & 0 & 0 \\
  p_3 & 0 & 2\alpha\beta & 0 & 0 & 0 \\
  p_4 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 
\end{bmatrix}.$$

Since $c_{20} = 1$ and $c_{02} = 0$, the discussion in Section 3.3 predicts

- 2 polynomials equal to $\alpha^2$, one equal to $2\alpha\beta$ and one equal to 0, or
- 2 polynomials equal to $\alpha^2$, one equal to $c_{11}\alpha\beta$ and one equal to 0, or
- 1 polynomial equal to $\alpha^2$, one equal to $\alpha^2 + c_{11}\alpha\beta$ and one equal to 0,

according to whether $x_i$ is an off-diagonal entry of $X$, $x_i$ is an off-diagonal entry of $Y$ or $x_i$ is a diagonal entry of $Y$. Thus, $c_{11} = 0$, $x_2$ and $x_3$ must belong to $X$, and $x_5, \ldots, x_8$ belong to $Y$.

Next, we shall arbitrarily place $x_2$ in the 12 spot of $X$. Then $x_3$ must be in the 21 spot of $X$, and $p_2$ is in the 12 spot of $p(X, Y)$. Now, having placed the entries in $X$ and positioned the polynomials $p_1, \ldots, p_4$ in $p(X, Y)$, it is readily seen that

$x_1x_6$ a term in $p_2$, $x_1$ in the 22 position of $X \implies x_6$ in the 12 position of $Y$

$x_2x_7$ a term in $p_1$, $x_2$ in the 12 position of $X \implies x_7$ in the 21 position of $Y$.

However, it is not possible to fix the positions of $x_5$ and $x_8$ within $Y$ from the available information. Thus, to this point we know that

$$X = \begin{bmatrix} x_4 & x_2 \\ x_3 & x_1 \end{bmatrix} \text{ and that either } Y = \begin{bmatrix} x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \text{ or } Y = \begin{bmatrix} x_8 & x_6 \\ x_7 & x_5 \end{bmatrix},$$

and, since $c_{20} = 1$ and $c_{11} = c_{02} = 0$, that there should be an nc polynomial of the form

$$p(X, Y) = X^2 + a(XY - YX)$$

for some $a \in \mathbb{R}$. Since the coefficients of all the terms in $p_1, \ldots, p_4$ are $\pm 1$, it follows that $a = \pm 1$. It is then readily checked that

$$\begin{bmatrix} p_4 & p_2 \\ p_3 & p_1 \end{bmatrix} = X^2 + X \begin{bmatrix} x_8 & x_6 \\ x_7 & x_5 \end{bmatrix} - \begin{bmatrix} x_8 & x_6 \\ x_7 & x_5 \end{bmatrix} X,$$

i.e., the second choice of $Y$ works, the first does not.
Example 5.4. Let \( k = 5 \) and \( n = 3 \) and suppose that \( x_i \) is in the \( ii \) position for \( i = 1, \ldots, 5 \) and that

\[
X = \begin{bmatrix}
x_1 & x_6 & x_8 & x_{10} & x_{12} \\
x_7 & x_2 & x_{14} & x_{16} & x_{18} \\
x_9 & x_{15} & x_3 & x_{20} & x_{22} \\
x_{11} & x_{17} & x_{21} & x_4 & x_{24} \\
x_{13} & x_{19} & x_{23} & x_{25} & x_5
\end{bmatrix}.
\]

The objective is to analyze the family of 25 commutative polynomials in 25 variables determined by \( X^3 \) to recreate \( X \). As in the previous examples, the family of polynomials is the given data; we only include \( X \) here because we will not write out the full family determined by \( X^3 \) due to its prohibitive size. The reader is encouraged to create the family generated by \( X^3 \) using Mathematica and then follow along in the analysis to recreate \( X \).

Discussion If \( X \) is of the given form, then \( X^3 \) generates a \( 5 \times 5 \) array of homogeneous polynomials of degree three, \( p_1, \ldots, p_{25} \). Five of these polynomials will each contain exactly one term of the form \( x_i^3 \). To simplify the exposition, we shall assume that the variables are indexed so that \( x_i \) is in the \( ii \) position of \( X \) for \( i = 1, \ldots, 5 \). This in turn forces the polynomial that contains \( x_i^3 \) to be in the \( ii \) position of \( X^3 \). The rest of the construction is broken into steps.

1. Find the entries in \( L_i = R_i \cup C_i \) for \( i = 1, \ldots, 5 \) by considering the terms \( x_i^2 x_j \) that appear in \( X^3 \), \( i = 1, \ldots, 5 \) and \( j = 6, \ldots, 25 \). For the given \( X \) we will obtain:

\[
\begin{align*}
L_1 &= \{x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}, \\
L_2 &= \{x_6, x_7, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}\}, \\
L_3 &= \{x_8, x_9, x_{14}, x_{15}, x_{20}, x_{21}, x_{22}, x_{23}\}, \\
L_4 &= \{x_{10}, x_{11}, x_{16}, x_{17}, x_{20}, x_{21}, x_{24}, x_{25}\}, \\
L_5 &= \{x_{12}, x_{13}, x_{18}, x_{19}, x_{22}, x_{23}, x_{24}, x_{25}\}
\end{align*}
\]

2. Position the entries in \( L_1 \).

Observe that \( L_1 \cap L_2 = \{x_6, x_7\} \). This means that one of these variables is in the 12 position and the other is in the 21 position. We shall assume that \( x_6 \) is in the 12 position and shall deduce the positions of all the other variables from the 25 polynomials \( p_1, \ldots, p_{25} \), corresponding to \( X^3 \). In particular, the assumption that \( x_6 \) is in the 12 position implies that the polynomial

\[
x_6^2 x_6 + x_6 x_2^2 + x_1 x_6 x_2 + x_1 x_8 x_{15} + x_1 x_{10} x_{17} + x_1 x_{12} x_{19} + \cdots
\]

is in the 12 position of \( X^3 \). But the remaining variables \( x_1 x_s x_t \) in that polynomial with off-diagonal variables \( x_s \) and \( x_t, s \neq t \), will be in the 12 position of \( X^3 \) if and only if they are positioned in one of the ways indicated in the following array:

\[
\begin{array}{cccccccc}
x_s & 13 & 14 & 15 & 32 & 42 & 52 \\
x_t & 32 & 42 & 52 & 13 & 14 & 15
\end{array}
\]
which is to be read as:

either \( x_s \) is in the 13 position and \( x_t \) is in the 32 position, or vice versa

The pair \( x_s = x_8 \) and \( x_t = x_{15} \) are subject to these constraints. On the other hand, since \( x_8 \in L_1 \cap L_3 \) it can only be in either the 13 position or the 31 position, whereas \( x_{15} \in L_2 \cap L_3 \) and hence can only be in the 23 position or the 32 position. Thus, the only viable solution to all these constraints is that

\[ x_8 \text{ is in the 13 position and } x_{15} \text{ is in the 32 position.} \]

Similarly, since \( x_1 x_{10} x_{17} \) and \( x_1 x_{12} x_{19} \) are in \( p_{12} \), whereas \( x_{10}, x_{17} \in L_1 \cap L_4 \), \( x_{12} \in L_1 \cap L_5 \) and \( x_{19}, x_{15} \in L_2 \cap L_5 \), it is readily seen that

\[ x_{10} \text{ is in the 14 position and } x_{17} \text{ is in the 42 position} \]

and

\[ x_{12} \text{ is in the 15 position and } x_{19} \text{ is in the 52 position.} \]

Moreover, now that \( x_6, x_8, x_{10} \) and \( x_{12} \) are positioned, we consider the following monomials in \( p_{12} \):

\[
\begin{align*}
x_6 x_8 x_9 & \implies x_9 \text{ is in the 31 position of } X \\
x_6 x_{10} x_{11} & \implies x_{11} \text{ is in the 41 position of } X \\
x_6 x_{12} x_{13} & \implies x_{13} \text{ is in the 51 position of } X.
\end{align*}
\]

**Remark 5.5.** The preceding calculations exploit the entries in the polynomial in the 12 position of \( p(X, Y) \) to calculate terms in \( R_1, C_1 \) and \( R_2 \). A variant of this is to just fill in \( R_1 \) and then to rely on successive steps to fill in \( R_2, \ldots, R_5 \), one row at a time. At the other extreme, it is also possible to use all the entries in this polynomial to fill in the whole matrix; see item 6, below.

3. Position the remaining entries in \( L_2 \) by the algorithm.

To this point we know that

\[
X = \begin{bmatrix}
x_1 & x_6 & x_8 & x_{10} & x_{12} \\
x_7 & x_2 & \cdot & \cdot & \cdot \\
x_9 & x_{15} & x_3 & \cdot & \cdot \\
x_{11} & x_{17} & \cdot & x_4 & \cdot \\
x_{13} & x_{19} & \cdot & \cdot & x_5
\end{bmatrix},
\]

and it remains to fill in the dots. Since the polynomial

\[ x_7 x_1^2 + x_2^2 x_7 + x_2 x_{14} x_9 + x_2 x_{16} x_{11} + x_2 x_{18} x_{13} + \cdots \]

sits in the 21 position of \( X^3 \) and the positions of \( x_9, x_{11} \) and \( x_{13} \) are known, it follows that

\[ x_{14} \text{ is in the 23 position, } x_{16} \text{ is in the 24 and } x_{18} \text{ in the 25.} \]

Moreover, since

\[ \{x_{14}, x_{15}\} = L_2 \cap L_3 \text{ and } x_{14} \text{ is in the 23 position} \]
it follows that \( x_{15} \) is in the 32 position. Similarly, since

\[ \{ x_{16}, x_{17} \} = L_2 \cap L_4 \quad \text{and} \quad x_{16} \text{ is in the 24 position} \]

it follows that \( x_{17} \) is in the 42 position. Much the same argument based on the position of \( x_{18} \) and the observation that \( L_2 \cap L_5 = \{ x_{18}, x_{19} \} \) shows that \( x_{19} \) is in the 52 position of \( X \). This completes the positioning of the variables in \( L_2 \).

4. Position the remaining entries in \( R_3 \) using Algorithm ParPosX outlined in Section 4.1

Since the polynomial

\[ x_3^2 x_9 + x_9 x_1^2 + x_3 x_{15} x_7 + x_3 x_{20} x_{11} + x_3 x_{22} x_{13} + \cdots \]

is in the 31 position of \( X^3 \) and the positions of \( x_3, x_7, x_{11} \) and \( x_{13} \) are known, it is readily checked that \( x_{20} \) is in the 34 position and \( x_{22} \) is in the 35 position.

5. Position the remaining entries in \( X \) using Algorithm ParPosX.

Since the polynomial

\[ x_4^2 x_{11} + x_{11} x_1^2 + x_4 x_{17} x_7 + x_4 x_{21} x_9 + x_4 x_{24} x_{13} + \cdots \]

is in the 41 position of \( X^3 \) and the positions of \( x_4, x_7, x_9 \) and \( x_{13} \) are known, it follows that \( x_{21} \) is in the 43 position and \( x_{24} \) is in the 45 position. Similarly, as the polynomial

\[ x_5^2 x_{13} + x_{13} x_1^2 + x_5 x_{19} x_7 + x_5 x_{23} x_9 + x_5 x_{25} x_{11} + \cdots \]

is in the 51 position of \( X^3 \), the positions of \( x_5, x_7, x_9 \) and \( x_{11} \) serve to locate \( x_{19} \) in the 52 position, \( x_{23} \) in the 53 position and \( x_{25} \) in the 54 position. This completes the computation via Algorithm ParPosX.

6. Variations on the theme.

The preceding steps fill in the entries of the partially specified matrix \( X \) in (5.1) by studying the entries in 5 of the given 25 polynomials by sweeping along rows. It is also possible to fill in \( X \) by using all the entries in the polynomial

\[ q = x_1^2 x_6 + x_6 x_2^2 + x_1 x_6 x_2 + x_1 x_8 x_{15} + x_1 x_{10} x_{17} + x_1 x_{12} x_{19} + \cdots \]
that sits in the 12 position of $X^3$:

- $x_6x_8x_9$: a term in $q$, $x_6$ in the 12 spot, $x_8$ in the 13 spot $\implies x_9$ in the 31 spot
- $x_6x_{10}x_{11}$: a term in $q$, $x_6$ in the 12 spot, $x_{10}$ in the 14 spot $\implies x_{11}$ in the 41 spot
- $x_6x_{12}x_{13}$: a term in $q$, $x_6$ in the 12 spot, $x_{12}$ in the 15 spot $\implies x_{13}$ in the 51 spot
- $x_6x_{14}x_{15}$: a term in $q$, $x_6$ in the 12 spot, $x_{15}$ in the 32 spot $\implies x_{14}$ in the 23 spot
- $x_6x_{16}x_{17}$: a term in $q$, $x_6$ in the 12 spot, $x_{17}$ in the 42 spot $\implies x_{16}$ in the 24 spot
- $x_6x_{18}x_{19}$: a term in $q$, $x_6$ in the 12 spot, $x_{19}$ in the 52 spot $\implies x_{18}$ in the 25 spot
- $x_8x_{20}x_{17}$: a term in $q$, $x_8$ in the 13 spot, $x_{17}$ in the 42 spot $\implies x_{20}$ in the 34 spot
- $x_8x_{22}x_{19}$: a term in $q$, $x_8$ in the 13 spot, $x_{19}$ in the 52 spot $\implies x_{22}$ in the 35 spot
- $x_{10}x_{15}x_{21}$: a term in $q$, $x_{10}$ in the 14 spot, $x_{15}$ in the 32 spot $\implies x_{21}$ in the 43 spot
- $x_{12}x_{15}x_{23}$: a term in $q$, $x_{12}$ in the 15 spot, $x_{15}$ in the 32 spot $\implies x_{23}$ in the 53 spot
- $x_{12}x_{17}x_{25}$: a term in $q$, $x_{12}$ in the 15 spot, $x_{17}$ in the 42 spot $\implies x_{25}$ in the 54 spot

Therefore, the single remaining variable $x_{24}$ must be in the 45 spot.

**Remark 5.6.** If, in Step 2, $x_6$ is placed in the 21 position and $x_7$ is placed in the 12 position, then the algorithm will generate the transpose $X^T$ of the matrix $X$ that is specified in the statement of Example 5.4.

### 6. Cost of Alg. 2 vs. a Brute Force Approach

Now that we have developed our one-letter methods for determining an nc representation and determined which families they will work for, we would like to get some idea of their cost. As a sample we get some rough cost estimates for Alg.2 and omit estimates for Alg.1. As Alg.1 and Alg.2 differ only in the partitioning algorithms used, we suspect that the costs for Alg.1 will be similar to the costs for Alg.2. The goal of determining these estimates is to compare the gains in efficiency that our methods provide over the more direct, brute force approach described in §1.6. As usual, we assume $P$ is a family containing $k^2$ polynomials of degree $d$ or less in the commutative variables $x_1, \cdots, x_{2k^2}$.


Recall that in Section 4.6 we determined the cost of solving the linear systems determined by NcCoef. As the Brute Force Method merely applies algorithm NcCoef for each arrangement of the commutative variables and polynomials, our work is nearly finished. However, recall that in our cost analysis of algorithm NcCoef we omitted the cost to form the systems. For the sake of completeness, we briefly mention how to form these linear systems using a function like the one in Mathematica called CoefficientList[] to get an idea of the cost.

The function CoefficientList[] in Mathematica takes as an input a polynomial and a collection of commutative variables and outputs the coefficients of the polynomial associated with the given variables. For example, the command

$$A[i] = \text{CoefficientList}[p_i, \{x_1, \cdots, x_{2k^2}\}]$$
creates a multidimensional array $A[i]$, where the entry $A[i](i_1, \ldots, i_{2k^2})$ corresponds to the coefficient of the monomial $x_1^{i_1} \cdots x_{2k^2}^{i_{2k^2}}$ in $p_i$. The starting point for our implementations of both the brute force approach and Alg.2 will be to call CoefficientList[]. In addition to this, recall that the cost to solve the linear systems formed by Eq. (6.1) by setting $1 \leq j \leq k^2$ for each $p_i$ of the given family $P$. We note that any implementation of our algorithms will require a function similar to CoefficientList[] in order to access the coefficients of the family $p_1, \ldots, p_{2k^2}$ in a systematic way.

Let $M_i$ be the system formed by equating

$$P^i_\lambda = p^i_\sigma(X, Y),$$

where $P^i_\lambda$ and $p^i_\sigma(X, Y)$ are formed from (4.11) and (4.12) respectively, by homogeneous sorting. Then let $p_{\lambda(j)}$ be an element of the matrix $P_\lambda$, and let $q_j$ be an element of $p_\sigma(X, Y)$ with undetermined coefficients $c_{\alpha \beta}$. Letting $A[j] = \text{CoefficientList}[q_j, \{x_1, \ldots, x_{2k^2}\}]$ for each $1 \leq j \leq k^2$, and $B[j] = \text{CoefficientList}[p_{\lambda(j)}, \{x_1, \ldots, x_{2k^2}\}]$, we obtain the system associated with Eq. (6.1) by setting

$$A[j](i_1, \ldots, i_{2k^2}) = B[j](i_1, \ldots, i_{2k^2}) \quad (1 \leq j \leq k^2)$$

for $(i_1, \ldots, i_{2k^2})$ satisfying $\sum_{n=1}^{2k^2} i_n = i$.

Excluding the cost of iterating through the terms in Eq. (6.2), the cost to form the systems of equations associated with NcCoef involves $k^2$ calls to the function CoefficientList[]. In addition to this, recall that the cost to solve the linear systems formed by NcCoef for a given arrangement $\sigma$ of the variables and $\lambda$ of the polynomials satisfies

$$\text{CostNcCoef}_{\sigma \lambda} \leq \sum_{i=2}^{d} \left( \tau_i A^i - \frac{8^i}{3} \right) \text{ arithmetic operations},$$

provided $\tau_i > 2^i$. The formula is similar if $\tau \leq 2^i$.

Recall that there are $(2k^2)!$ arrangements $\sigma$ of $x_1, \ldots, x_{2k^2}$ in $X$ and $Y$ and $(k^2)!$ arrangements $\lambda$ of $p_1, \ldots, p_{2k^2}$. We note that we must call CoefficientList[] once for each $p_j$ in the given family $P$ and once for each undetermined polynomial $q_j$ determined by $\sigma$. Therefore if $\tau > 2^i$, the cost satisfies

$$\text{CostBruteForce} \leq (2k^2)!(k^2)! \left( \sum_{i=2}^{d} \left( \tau_i A^i - \frac{8^i}{3} \right) \right) \text{ arithmetic operations}$$

$$+ \left( (k^2 + (2k^2)!) \text{ Calls to CoefficientList[]} \right).$$

6.2. Cost of Algorithm 2. In this section we roughly analyze the cost associated with each subroutine required to execute Alg.2. Then in Section 6.2.7 we summarize these costs and compare them to the brute force cost determined in Eq (6.4).

6.2.1. Algorithm DiagA: Determining the Diagonal Entries. The first step in Alg.2 is to determine the commutative variables that occur in one-letter monomials of the form $ax_{i_1}^i$. Here we give a rough outline of a procedure based on Lemma 2.7 that will do this, and then analyze the cost.

As in the case of the brute force algorithm and algorithm NcCoef, we implement this algorithm using the function CoefficientList[]. Again, we let

$$A[i] = \text{CoefficientList}[p_i, \{x_1, \ldots, x_{2k^2}\}].$$
Using the coefficient matrix $A[i]$ for each $1 \leq i \leq k^2$, we inspect the coefficients of the single letter monomials of degree $n$, where $2 \leq n \leq d$. For example, the coefficients of the one letter monomials $ax_1^n$ and $bx_2^n$ in $p_i$ are given by

$$A[i](n, 0, \cdots, 0) = a \quad \text{and} \quad A[i](0, n, \cdots, 0) = b.$$ 

The core of Algorithm DiagA is to:

Fix a degree $2 \leq n \leq d$. Then iterate through all of the coefficients of one letter monomials of degree $n$ in $\mathcal{P}$. That is, iterate through the expressions

$$(6.5) \quad A[1](n, 0, \cdots, 0), \cdots, A[1](0, \cdots, 0, n)$$
$$A[2](n, 0, \cdots, 0), \cdots, A[2](0, \cdots, 0, n)$$
$$\vdots$$
$$A[k^2](n, 0, \cdots, 0), \cdots, A[k^2](0, \cdots, 0, n)$$

and verify that exactly $0, k$ or $2k$ of the above expressions are nonzero and that they assume at most two distinct, nonzero values. If this is satisfied, then store the nonzero coefficients and store the associated diagonal variable in one of two lists based on its coefficient. If this is not satisfied then quit.

6.2.2. Cost Estimate for DiagA. The cost of algorithm DiagA is easy to bound and can be expressed in terms of the parameters $k$ and $d$ and the number of calls to CoefficientList[]. Define an equality check to be a test if two given numbers are equal.

Algorithm DiagA begins with $k^2$ calls to the function CoefficientList[], one for each polynomial $p_i$. Iterating through the $(2k^2)(k^2)$ coefficients listed in (6.5) for a given degree $2 \leq n \leq d$ and performing a small number $C_A$ of equality checks for each coefficient requires no more than $(d-1)(k^2)(2C_Ak^2) = 2C_A(d-1)k^4$ equality checks. Therefore, the total cost for applying DiagA is

$$(6.6) \quad TC = (C_A(d-1)k^4 \text{ equality checks}) + (k^2 \text{ Calls to CoefficientList[]}) .$$

Here $C_A$ is certainly no greater than 5.

6.2.3. Cost of DiagPar2. Here we determine an upper bound for the cost of Algorithm DiagPar2. We assume that DiagA has already been executed and that CoefficientList[] has been called for each polynomial $p_i \in \mathcal{P}$. See §6.2.1 for a definition of these procedures.

Algorithm DiagPar2 utilizes the coefficients of monomials of the form $ax_i^{n-1}x_j$, where $x_i$ is a diagonal variable and $x_i \neq x_j$, to partition the variables between $X$ and $Y$. The bulk of the cost of this algorithm will lie in iterating through the coefficients of the two-letter monomials of this form. We fix a degree $2 \leq n \leq d$, a polynomial $p_j$ and a diagonal element $x_i$ (which can be determined using DiagA) and iterate through the coefficients of monomials in $p_j$ of the form $x_i^{n-1}x_l$, where $x_l$ is a non-diagonal element; therefore, we iterate through $2k^2 - 2$ coefficients for each diagonal element. For each coefficient we perform no more than $C_{Par2}$, a small number, of equality checks to
ensure that the conditions in Lemma 2.12 are satisfied and to determine whether \( x_l \) is in \( X \) or \( Y \).

If at any point we find that these conditions are violated or we cannot find an \( n \) such that the two-letter monomials \( ax_i^{n-1}x_j \) satisfy the conditions consistent with (3.10), we terminate the iteration and conclude either that \( \mathcal{P} \) has no nc representation or that the algorithm is inconclusive. As we are only considering \( k \) diagonal variables, \( k^2 \) polynomials and \( d - 1 \) different degrees that need to be considered, we have that
\[
\text{TotCostDiagPar2} \leq C_{Par} (d - 1)k^5 \text{ equality checks.}
\]

Here \( C_{Par} \leq 5 \).

6.2.4. Cost of ParPosX. We now determine a rough bound on the cost of Algorithm ParPosX. Again, we assume Algorithm DiagA and DiagPar2 have been executed and that CoefficientList[] has been called for each of the polynomials \( p_i \in \mathcal{P} \).

Algorithm ParPosX positions the variables in \( X \) by using lists \( L_i \) defined in (4.1) and three letter monomials of the form \( x_i x_l x_m \), where \( x_i \) is a diagonal and \( x_l \neq x_m \) are non-diagonal elements in \( X \). We observe that we can build the lists \( L_i \) when we iterate through the two letter monomials \( ax_i^{n-1}x_j \in \text{DiagPar2} \), so other than the cost of DiagA, DiagPar1 or DiagPar2, the primary cost associated with this algorithm lies in iterating through the three-letter monomials. For a given non-diagonal polynomial \( p_j \) and diagonal variable \( x_i \), we must iterate through the coefficients of \((k^2 - k)(k^2 - k - 1)\) such monomials, and for each nonzero coefficient (of which there will be at most \( k - 1 \)), iterate through a list of length \( 2k - 2 \) and perform a fixed number of equality checks which we will designate by \( C_{ParP} \). Therefore,
\[
\text{TotCostPar2Pos} = C_{ParP}(k^2 - k)k[((k^2 - k)(k^2 - k - 1)) - k] + C_{ParP}(k^2 - k)k(k - 1)(2k - 2) \text{ equality checks} \\
\leq 3C_{ParP}k^7 \text{ equality checks.}
\]

Here \( C_{ParP} \leq 10 \).

6.2.5. Cost of PosPol. As in the case of Algorithm DiagPar2, the cost of Algorithm PosPol is effectively reduced to the cost of iterating through two-letter coefficients. As in DiagPar2, we inspect two-letter monomials of the form \( ax_i^{n-1}x_j \), where \( x_i \) is a diagonal element of \( X \) and \( x_j \) is a non-diagonal element of \( X \). For a fixed degree \( 2 \leq n \leq d \), polynomial \( p_m \) and diagonal element \( x_i \) of \( X \), we iterate through the coefficients of all possible two-letter monomials in \( p_m \), which allows for \( k^2 - k \) possibilities. Once we find a coefficient \( a \neq 0 \) corresponding to the monomial \( ax_i^{n-1}x_j \), we immediately determine the position of \( p_m \) using the position of \( x_i \) and \( x_j \). Therefore
\[
\text{TotalCostPolyPos} \leq C_{PosP}(d - 1)k^5 \text{ equality checks, } C_{PosP} \leq 10.
\]

We note that this is probably a gross overestimate of the cost of PosPol and that one does not need to iterate through all of the two-letter monomials again. It seems possible that we could store additional data associated with the non-diagonal entries \( x_j \)
when we iterate through the two-letter algorithms in DiagPar2 in order to position the polynomials.

6.2.6. Cost of PosY. We now determine a bound for the cost of PosY for families $\mathcal{P} \in NC_{3.11}$. We again assume that all necessary arguments to implement PosY have been executed so that the commutative variables are positioned in $X$ and the polynomials are positioned. Therefore we assume that half of the commutative variables have been positioned in $X$ and that the polynomials $p_1, \ldots, p_{k^2}$ have been assigned positions in a $k \times k$ matrix. The goal is to to position the remaining variables in $Y$.

The first step is to determine the diagonal variables of $Y$. PosY uses the fact that $\varphi(s, t) \neq 0$ for some $s, t$ such that $2 \leq s + t = n \leq d$ and $t > 0$. Then we look at two letter monomials of the form $\varphi(s, t)x_i^s x_l^t$ in the diagonal polynomials $p_j$, where $x_i$ is a diagonal variable of $X$. This will allow us to conclude that $x_i$ is a diagonal variable in $Y$ that must be located in the same position as $x_i$ is in $X$. To implement this step, for a fixed degree $n$, diagonal polynomial $p_j$ and corresponding diagonal variable $x_i$ in $X$, and $s, t$ satisfying $s + t = n$, we must iterate through $k^2 - k$ coefficients corresponding to two letter monomials of the form $x_i^s x_l^t$, where $x_i$ is in $Y$. Therefore

\begin{equation}
\text{CostDiagY} \leq C_{\text{PosY}} (d - 1)(k) \left( \frac{d}{2} \right) (k^2 - k) \text{ equality checks}
\leq C_{\text{PosY}} (d^3 k^3) \text{ equality checks},
\end{equation}

where $C_{\text{PosY}}$ dominates the operation per step count.

Finally, to determine the position of the off-diagonal variables of $Y$, we iterate through three-letter monomials of the form $ax_i^s x_l^t x_v$, where $x_i$ and $x_l$ are diagonal variables of $X$ and $Y$ respectively, and $x_v$ is a non-diagonal term of $Y$. We observe that if such a term occurs in a polynomial $p_j$ that is in the $qr$-position of the polynomial matrix and $x_i$ and $x_l$ are in the $qq$ position of $X$ and $Y$ respectively, then $x_v$ must be in the $qr$-position of $Y$. Again, the bulk of the cost lies in iterating through these terms. For a fixed non-diagonal polynomial $p_j$ and fixed diagonal variables $x_i$ and $x_l$ in the $qq$-position in $X$ and $Y$, we consider the coefficients of $k^2 - k$ such terms and fewer than $C_{\text{PosY}}$ equality checks each. Therefore,

\begin{equation}
\text{CostNonDiagY} \leq C_{\text{PosY}} (k^2 - k)(k^2 - k) \text{ equality checks}
\leq C_{\text{PosY}} (k^5) \text{ equality checks}.
\end{equation}

Thus,

\begin{equation}
\text{TotalCostPosY} \leq C_{\text{PosY}} \left( (d^3 k^3) + (k^5) \right) \text{ equality checks}
\end{equation}

Here $C_{\text{PosY}} \leq 10$.

6.2.7. Cost of Algorithm 2. The following table lists the subroutines that make up Alg.2, their cost in terms of the number of operations and calls to CoefficientList[], and the section in which the cost of the subroutines was determined. Recall that the parameter $\tau_i$ represents the total number of commutative monomials of degree $2 \leq i \leq d$ in all polynomials in $\mathcal{P}$, and $C_{\text{DiagA}}, C_{\text{Par2}}, C_{\text{ParP}}, C_{\text{PosP}}, C_{\text{PosY}}$ are all constants bounded by 10.
Algorithm | Operations | Calls to CoefficientList[]
--- | --- | ---
DiagA | $C_{\text{DiagA}} dk^4$ | $k^4$
DiagPar2 | $C_{\text{DiagPar2}} dk^5$ | 0
ParPosX | $C_{\text{ParPosX}} k^i$ | 0
PosPol | $C_{\text{PosPol}} dk^4$ | 0
PosY | $C_{\text{PosY}} (d^3k^5 + k^3)$ | 0

Combining this with the cost of applying algorithm NcCoef described in §4.6, we obtain

\begin{equation}
\text{CostAlg2} \leq 10 \left( k^7 + (2d + 1)k^5 + dk^4 + d^3k^3 \right) \text{ equality checks} \\
+ \sum_{i=2}^{d} \left( \tau_i 4^i - \frac{8i}{3} \right) \text{ arithmetic operations} + k^2 \text{ Calls to CoefficientList[]},
\end{equation}

provided $\tau_i > 2^i$ for each $i$. When $\tau_i \leq 2^i$ for each $i$, we get

\begin{equation}
\text{CostAlg2} \leq 10 \left( k^7 + (2d + 1)k^5 + dk^4 + d^3k^3 \right) \text{ equality checks} \\
+ \sum_{i=2}^{d} \frac{2^{3i+1}}{3} \text{ arithmetic operations} + k^2 \text{ Calls to CoefficientList[]}.
\end{equation}

6.2.8. Comparison of Costs. A comparison of the bounds (6.4),(6.15) and (6.14) shows the benefit of our Algorithms. We first observe that we need to use the function CoefficientList[] $(2k^2)!$ fewer times using Alg.2, which is a huge savings given that the cost to use CoefficientList[] could potentially be very expensive. Even if we neglect the cost of this function, we see that for large $d$ and $k$, the cost to form and solve linear systems using NcCoef dominates both the cost for the Brute Force Method and Alg.2. By exploiting the structure of the polynomials in the given family $\mathcal{P}$ and performing on the order of

\[ k^7 + 3dk^5 + d^3k^3 \text{ operations}, \]

we have effectively reduced the cost from solving $(2k^2)!(k^2)!$ such systems to a single system. This is a vast improvement. Also, to rule out the existence of an nc representation using the Brute Force method we must check all of these cases and verify that they fail. Much to the contrary, Algorithm 2 is likely to determine non existence very early when applying it.

7. Families that may not contain a term of the form $ax^n_i$ with $n > 1$

Sections 3, 4 and 6 present algorithms for solving our nc representation problem when at least one of the given polynomials in $\mathcal{P}$ contains a term of the form $ax^n_i$ with $a \in \mathbb{R} \setminus \{0\}$ and $n > 1$. This section treats $\mathcal{P}$ which contain no one letter monomials but which do contain terms of the form $ax^n_i x^t_j$, $a \neq 0$. Recall Lemma 2.8 and Lemma 2.10 dealt with patterns two letter monomials in an nc representable $\mathcal{P}$ must obey.
The first step in developing these two letter algorithms is to determine the diagonal elements given the existence of two letter monomials. Lemmas 2.16 and 2.17 present conditions under which the presence of terms that are $\nabla$-equivalent to $x_i^s x_j^t$ allows us to determine dyslexic diagonal pairs. The next step is to partition the $k$ dyslexic pairs \{\(x_{i1}, x_{j1}\), ..., \(x_{ik}, x_{jk}\), i.e., to determine which elements are on the diagonal of $X$ and which are on the diagonal of $Y$. An application of Lemma 2.19 is usually sufficient to partition these diagonal pairs.

Once the diagonal variables are determined and partitioned, we can reduce the problem to one that is manageable for the single variable algorithms by taking derivatives of the polynomials in $P$ with respect to the diagonal variables. This process will be known as the SVR algorithm. The next section discusses this reduction.

We shall not present a cost analysis of our two-letter methods in this section, since one can see as they unfold that they are clearly far superior to Brute Force. For one thing the core of our two-letter procedures are reductions to our single letter algorithms (such as Alg. 2 whose cost is vastly less than of Brute Force).

7.1. Reduction to one letter algorithms: SVR Algorithm. In this section we develop the SVR (Single Variable Reduction) Algorithm, which can be used to reduce a family of polynomials containing terms that are $\nabla$-equivalent to $x_i^s x_j^t$ to a family of polynomials to which the single variable algorithms developed earlier apply. The next lemma contains the key idea that underlies this algorithm.

**Lemma 7.1.** If $p_1, \ldots, p_{k^2}$ is a family of polynomials in the $2k^2$ commuting variables $x_1, \ldots, x_{2k^2}$ that admits an nc representation $p(X,Y)$ of degree $d \geq 2$ such that $x_{ir}$ is in the $ab$ position of $X$ and $x_{i1}, \ldots, x_{ik}$ sit on the diagonal of $X$, then:

1. The polynomials
   \[
   \frac{\partial p_1}{\partial x_{ir}}, \ldots, \frac{\partial p_{k^2}}{\partial x_{ir}}
   \]
   admit the nc representation
   \[
   \lim_{t \to 0} \frac{1}{t} (p(X + tE_{ab}, Y) - p(X, Y)).
   \]
   \[
   (7.1)
   \]

2. The family of polynomials $q_m(x_1, \ldots, x_{k^2}), m = 1, \ldots, k^2$, defined by the formula
   \[
   q_m(x_1, \ldots, x_{2k^2}) = \sum_{s=1}^k \frac{\partial}{\partial x_{is}} p_m(x_1, \ldots, x_{2k^2}), \quad m = 1, \ldots, k^2,
   \]
   admits the nc representation
   \[
   \lim_{t \to 0} \frac{1}{t} (p(X + tI_k, Y) - p(X, Y)).
   \]
   \[
   (7.2)
   \]

**Proof.** The first assertion follows from the observation that

\[
\lim_{t \to 0} \frac{1}{t} (p(X + tE_{ab}, Y) - p(X, Y))
\]
is equivalent to computing

\[
\lim_{t \to 0} \frac{p_m(x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r} + t, x_{i_{r+1}}, \ldots, x_{i_{2k^2}}) - p_m(x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r}, x_{i_{r+1}}, \ldots, x_{i_{2k^2}})}{t}
= \frac{\partial}{\partial x_{i_r}} p_m(x_{i_1}, \ldots, x_{i_{2k^2}}).
\]

The second assertion follows from the first by a straightforward calculation.

To ease future applications of Lemma 7.1 when \(x_{i_1}, \ldots, x_{i_k}\) are the diagonal elements of \(X\), it is convenient to introduce the notation

\[
(T_X p_m)(x_1, \ldots, x_{2k^2}) = \sum_{s=1}^{k} \frac{\partial}{\partial x_{i_s}} p_m(x_1, \ldots, x_{2k^2}), \quad m = 1, \ldots, k^2.
\]

**Remark 7.2.** Repeated application of the formulas in Lemma 7.1 serves to reduce nc expressions in \(X\) and \(Y\) to expressions in the single variable \(Y\). Thus for example, if \(p_1, \ldots, p_{k^2}\) is a family of polynomials corresponding to

\[
p(X, Y) = XY^2X^2,
\]

then the family of polynomials \(T^3 p_m, m = 1, \ldots, k^2\), corresponds to the polynomial

\[
3! p(I, Y) = 3! Y^2.
\]

In this way it is possible to eliminate the dependence on the \(k^2\) variables in \(X\) by differentiating the given family of polynomials just with respect to the \(k\) diagonal entries of \(X\).

Let \(D_{X,I_k} p(X, Y)\) (resp., \(D_{Y,I_k} p(X, Y)\)) denote the directional derivative of \(p(X, Y)\) with respect to \(X\) (resp., \(Y\)) in the direction of the identity \(I_k\). More generally, let \(D_{X,i_k}^i p(X, Y)\) (resp., \(D_{Y,i_k}^i p(X, Y)\)) denote the \(i\)-th directional derivative of \(p(X, Y)\) with respect to \(X\) (resp., \(Y\)) in the direction of \(I_k\). Then Lemma 7.1 states that if the variables \(x_{i_1}, \ldots, x_{i_k}\) are the diagonal elements of \(X\) (resp., \(Y\)), then \(D_{X,I_k} p(X, Y)\) (resp., \(D_{Y,I_k} p(X, Y)\)) is a representation of the family

\[
g_n = T_X p_n \quad \text{(resp. } g_n = T_Y p_n),
\]

where \(T_X\) (resp., \(T_Y\)) is the operator defined in (7.3).

**Remark 7.3.** By repeated application of Lemma 7.1, we have that \(D_{X,I_k}^i p(X, Y)\) (resp., \(D_{Y,I_k}^i p(X, Y)\)) is an nc representation of the family defined by

\[
g_n = T_X^i p_n \quad \text{(resp. } g_n = T_Y^i p_n)
\]

**Theorem 7.4 (SVR Algorithm).** Let \(p_1, \ldots, p_{k^2}\) be a homogeneous family of polynomials with an nc representation \(p(X, Y)\) of degree \(i + j\) where \(i \geq 2\) and \(j \geq 2\). Suppose that the \(k\) partitioned diagonal pairs

\[
\{x_{i_1}, x_{j_1}\}, \ldots, \{x_{i_k}, x_{j_k}\}
\]
are known and $\varphi(i, j) \neq 0$. Also assume that
\begin{equation}
\psi(i, j) \neq (i + 1)\varphi(i + 1, j - 1)
\end{equation}

or
\begin{equation}
\varphi(i, j) \neq (j + 1)\varphi(i - 1, j + 1).
\end{equation}

Then if (7.4) (resp., (7.5)) holds the variables can be partitioned by Algorithm DiagPar1 and then positioned in $Y$ (resp., $X$) by Algorithm ParPosX applied to $D_{X, I_k}^i p(X, Y)$ (resp., $D_{Y, I_k}^j p(X, Y)$).

**Proof.** Since the diagonal pairs are partitioned, we may assume that $x_{i_1}, \ldots, x_{i_k}$ are the diagonal entries of $X$ and $x_{j_1}, \ldots, x_{j_k}$ are the diagonal entries of $Y$. Moreover, the nc polynomial
\[ p(X, Y) = \sum_{s=0}^{i+j} q_s(X, Y), \]
where $q_s(X, Y)$ denotes the sum of the terms in $p(X, Y)$ that are of degree $s$ in $X$ and degree $i + j - s$ in $Y$. Consequently,
\[ (D_{X, I_k}^i p)(X, Y) = \sum_{s=0}^{i+j} (D_{X, I_k}^i q_s)(X, Y) \]
\begin{equation}
= \varphi(i, j)i!Y^j + (D_{X, I_k}^i q_{i+1})(X, Y) + \sum_{s=i+2}^{i+j} (D_{X, I_k}^i q_s)(X, Y).
\end{equation}

Only the first two terms on the right in the second line of (7.6) will contribute two letter monomials of the form $ax_j^{j-1}x_u$. If $X$ is replaced by $A$, $Y$ is replaced by $B$ and $AB = BA$, then these two terms will be equal to
\[ i! \varphi(i, j)B^j + (i + 1)!\varphi(i + 1, j - 1)AB^{j-1} \]

Therefore, if $j(i!\varphi(i, j)) \neq (i + 1)!\varphi(i + 1, j - 1)$, then we may apply DiagPar1 to partition the variables. But this is the same as (7.4). Moreover, in view of Theorem 4.1, Algorithm ParPosX, will then serve to position the variables in $Y$.

Similarly, if (7.5) is in force, then DiagPar1 and ParPosX applied to $D_{Y, I_k}^j$ will serve to partition the variables and to position them in $X$. \qed

**Remark 7.5.** Theorem 7.4 remains valid if (7.4) and (7.5) are replaced by the conditions

$\varphi(i, j) \neq (i + 1)\varphi(i + 1, j - 1; X)$ or $\varphi(i, j) \neq (i + 1)\varphi(X; i + 1, j - 1).$

and

$\varphi(i, j) \neq (j + 1)\varphi(i - 1, j + 1; Y)$ or $\varphi(i, j) \neq (j + 1)\varphi(Y; i - 1, j + 1),$

respectively, but with DiagPar2 in place of DiagPar1. (In the first case, DiagPar2 is applied to $D_{X, I_k}^i p$, in the second, it is applied to $D_{Y, I_k}^j p$.)

The algorithm outlined in Theorem 7.4 will be called the **SVR** (Single Variable Reduction) Algorithm.
7.2. **Algorithms based on two letter words.** In this section we consider algorithms for families of polynomials $\mathcal{P}$ that contain two letter words that are $\triangleright$-equivalent to $x_i^s x_j^t$ with $i \neq j$ and $s \geq t \geq 2$. It is convenient to separately analyze the three mutually exclusive cases

$$s \geq t + 2, \quad s = t + 1, \quad \text{and} \quad s = t.$$ 

Our approach is to use either Alg.1 or Alg.2 in combination with the SVR algorithm. Recall that Alg.1 refers to the sequential application of DiagPar1, ParPosX, PosPol and PosY, whereas Alg.2 refers to the sequential application of DiagPar2, ParPosX, PosPol and PosY.

7.2.1. **Two letter monomials $\triangleright$-equivalent to $x^s u x^t v$ with $s \geq t + 2$.** This subsection contains two results that provide nc representations for families containing two letter monomials with $s \geq t + 2 \geq 4$; the first is based on Alg.1, the second on Alg.2.

Let $NC(7.7)$ denote the class of all nc polynomials $p(X,Y)$ of degree $d$ such that for some set of integers $s,t$ with

\begin{equation}
(7.7) \quad \left\{ \begin{array}{l}
s \geq t + 2 \geq 4, \quad \varphi(s,t) \neq 0, \quad \varphi(s,t) \neq \varphi(t,s) \\
t \varphi(s,t) \neq (s + 1) \varphi(s + 1, t - 1) \quad \text{or} \quad s \varphi(s,t) \neq (t + 1) \varphi(s - 1, t + 1)
\end{array} \right. \quad \text{and either}
\end{equation}

**Proposition 7.6.** Let $p_1, \ldots, p_{k^2}$ be a family $\mathcal{P}$ of polynomials in $2k^2$ commuting variables. Then the SVR algorithm coupled with Alg.1 will yield an nc representation of $p(X,Y)$ of $\mathcal{P}$ with $p$ in $NC(7.7)$ if and only if the given family $\mathcal{P}$ admits an nc representation in this set.

**Proof.** In view of the assumptions in the first line of (7.7), Lemma 2.16 may be applied to obtain the partitioned diagonal pairs. The constraints in the second line of (7.7) then insure that SVR algorithm coupled with Alg.1 serves to partition and position the variables and to position the given set of polynomials. More precisely, if $t \varphi(s,t) \neq (s + 1) \varphi(s + 1, t - 1)$ the differentiation is with respect to $X$; if $s \varphi(s,t) \neq (t + 1) \varphi(s - 1, t + 1)$, then the differentiation is with respect to $Y$. Moreover, if the differentiation in the SVR algorithm is with respect to $X$, then the ParPosX Algorithm will position the variables in $Y$ and the PosPol Algorithm will position the polynomials. On the other hand, if the differentiation in the SVR algorithm is with respect to $Y$, then the ParPosX Algorithm will position the variables in $X$. Finally, if the given family does not admit an nc representation, then the indicated algorithms cannot produce it. \hfill \Box

Let $NC(7.8)$ denote the class of all nc polynomials $p(X,Y)$ of degree $d$ such that for some set of integers $s,t$ with

\begin{equation}
(7.8) \quad \left\{ \begin{array}{l}
s \geq t + 2 \geq 4, \quad \varphi(s,t) \neq 0, \quad \varphi(s,t) \neq \varphi(t,s) \\
\varphi(s,t) \neq (s + 1) \varphi(s + 1, t - 1; X) \text{ or } \varphi(s,t) \neq (s + 1) \varphi(X; s + 1, t - 1) \\
or \\
\varphi(s,t) \neq (t + 1) \varphi(s - 1, t + 1; Y) \text{ or } \varphi(s,t) \neq (t + 1) \varphi(Y; s - 1, t + 1)
\end{array} \right.
\end{equation}
Proposition 7.7. Let $p_1, \ldots, p_{k^2}$ be a family $P$ of polynomials in $2k^2$ commuting variables. Then the SVR algorithm coupled with Alg.2 will yield an nc representation of $p(X,Y)$ of $P$ with $p$ in $NC(7.8)$ if and only if the given family $P$ admits an nc representation in this set.

Proof. The proof is similar to the proof of Proposition 7.6, except that Alg.2 is used in place of Alg.1. □

7.2.2. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s = t + 1$. The result for this case is simpler than the case when $s \geq t + 2 \geq 4$ because the condition

\[(7.9) \quad s \varphi(s,t) \neq (t+1)\varphi(s-1,t+1)\]

required for Alg.1 to work after an application of the SVR algorithm reduces to

\[\varphi(t+1,t) \neq \varphi(t,t+1)\]

when $s = t + 1$. However, we always require that $\varphi(t+1,t) \neq \varphi(t,t+1)$ so that we can successfully partition the diagonal variables. Therefore our conditions to insure that we can successfully partition the dyslexic diagonal pairs imply that Alg.1 will always be successful.

Let $NC(7.10)$ denote the class of all nc polynomials $p(X,Y)$ of degree $d$ such that for some integer $t$ with

\[(7.10) \quad \begin{cases} t \geq 2, & \varphi(t+1,t) \neq 0, \quad \varphi(t+1,t) \neq \varphi(t,t+1) \\ \varphi(t+1,t) \neq \text{the coefficient of } (XY)^tX \text{ or the coefficient of } (YX)^tY. \end{cases}\]

Proposition 7.8. Let $p_1, \ldots, p_{k^2}$ be a family $P$ of polynomials in $2k^2$ commuting variables. Then the SVR algorithm coupled with Alg.1 will yield an nc representation of $p(X,Y)$ of $P$ with $p$ in $NC(7.10)$ if and only if the given family $P$ admits an nc representation in this set.

Proof. The assumption that $\varphi(t+1,t) \neq 0, \varphi(t+1,t) \neq f_1, \varphi(t+1,t) \neq f_2,$ and $\varphi(t+1,t) \neq \varphi(t,t+1)$ allow Lemma 2.17 to determine the partitioned diagonal pairs. The SVR Algorithm can then be employed. The above discussion implies that if we differentiate with respect to $Y$ in the SVR Algorithm that the DiagPar1 Algorithm will successfully partition the remaining variables between $X$ and $Y$. Furthermore, the ParPosX Algorithm will position $X$ and once $X$ is determined, Algorithm PosPol positions the polynomials in the reduced family, which positions the polynomials in $p(X,Y)$. Once the polynomials are positioned, then Algorithm PosY positions $Y$ and Algorithm NcCoef will successfully determine $p(X,Y)$. □

Remark 7.9. We do not write out an analogous result based on Alg.2 because in order to partition the diagonal elements in Alg.2 we require that $\varphi(t+1,t) \neq \varphi(t,t+1)$, which insures that Alg.1 will be successful.

7.2.3. Two letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s = t$. The preceding two cases dealt with families of polynomials containing two-letter monomials $\triangleright$-equivalent to $x_u^s x_v^t$ with $s > t \geq 2$. In those cases Lemma 2.16 or Lemma 2.17 was applied first to determine the dyslexic diagonal pairs. Then Lemma 2.19 was applied to partition
a viable possibility. If \( \phi x \) with \( s = t \), then Lemma 2.19 is not applicable. This section is devoted to developing an algorithm for partitioning the diagonal entries and determining an nc representation in this particular case. The main result of this subsection is Theorem 7.11. It is convenient, however, to first establish a preliminary lemma:

**Lemma 7.10.** Let \( p_1, \ldots, p_{k^2} \) be a family of polynomials in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) with an nc representation \( p(X, Y) \). Let \( x_i \) and \( x_j \) be a dyslexic diagonal pair and let \( s_1, t_1 \) and \( s_2, t_2 \) be pairs of positive integers such that \( s_1 + t_1 = s_2 + t_2 \) and either \( s_1 = s_2 \) or \( s_1 + 1 = s_2 \) and suppose that \( p(X, Y) \) satisfies the following conditions when \( s + t = s_1 + t_1 \):

1. If \( s > 0 \), then \( \varphi(s, t + 1; X) \neq 0 \) if and only if \( \varphi(s, t + 1; Y) \neq 0 \).
2. If \( s > 0 \), then \( \varphi(X; s, t + 1) \neq 0 \) if and only if \( \varphi(Y; s, t + 1) \neq 0 \).
3. \( \varphi(0, s + t + 1) = \varphi(s + t + 1, 0) = 0 \).

Suppose further that \( \alpha x_i^{s_1} x_j^{t_1} x_u \) and \( \beta x_i^{s_2} x_j^{t_2} x_v \) appear in some polynomial \( p_r \in \mathcal{P} \) where \( \alpha \neq 0 \) and \( \beta \neq 0 \) and let \( \ell \) and \( m \) be the largest integers such that \( x_i^{s_1 - \ell} x_j^{t_1 + \ell} x_u \) and \( x_i^{s_2 - m} x_j^{t_2 + m} x_v \) appear in \( p_r \). Then

\[
\ell \geq m \implies x_u \in L(x_i) \text{ and } x_v \in L(x_j)
\]

and

\[
\ell < m \implies x_v \in L(x_i) \text{ and } x_u \in L(x_j).
\]

**Proof.** There are two steps:

1. \( \ell \geq m \implies x_u \in L(x_i) \): If \( \ell \geq m \) and \( x_u \not\in L(x_i) \), then \( x_u \) must belong to \( L(x_j) \), i.e., either \( x_u \in R(x_j) \) or \( x_u \in C(x_j) \). But if \( x_u \in R(x_j) \) and \( x_i \) and \( x_j \) are in the \( ss \) position of \( X \) and \( Y \), respectively, then \( x_u \) is in the \( st \) position of \( Y \) and \( x_v \) is in the \( st \) position of \( X \) (as \( \alpha x_i^{s_1} x_j^{t_1} x_u \) and \( \beta x_i^{s_2} x_j^{t_2} x_v \) are in the same polynomial \( p_r \)). Therefore, \( \varphi(s_1 - \ell, t_1 + \ell + 1; Y) \neq 0 \). If \( s_1 = \ell \), then this contradicts (3) and therefore is not a viable possibility. If \( s_1 > \ell \), then \( \varphi(s_1 - \ell, t_1 + \ell + 1; Y) \neq 0 \) and, (1) implies that \( \delta = \varphi(s_1 - \ell, t_1 + \ell + 1; X) \neq 0 \). Thus, \( \delta x_i^{s_1 - (\ell + 1)} x_j^{t_1 + \ell + 1} x_v \) belongs to \( p_r \), i.e.,

\[
\begin{align*}
\ell &= m \implies x_u \in L(x_i) \text{ and } x_v \in L(x_j)
\end{align*}
\]

However, the definition of \( m \) implies that \( \ell + 1 \leq m \) in the first case and \( \ell + 2 \leq m \) in the second, both of which clearly contradict the assumption that \( \ell \geq m \). Therefore, \( x_u \not\in R(x_j) \).

A similar argument based on (2) serves to prove that \( x_u \not\in C(x_j) \). Therefore, \( x_u \in L(x_i) \) as claimed. This completes the proof of 1.

2. \( m > \ell \implies x_v \in L(x_j) \): If \( m > \ell \) and \( x_v \not\in L(x_i) \), then \( x_v \in L(x_j) \). Suppose that in fact \( x_v \in R(x_j) \). Then \( \varphi(s_2 - m, t_2 + m + 1; Y) \neq 0 \). If \( s_2 = m \), then this contradicts (3); if \( s_2 > m \), then \( \varphi(s_2 - m, t_2 + m + 1; Y) \neq 0 \) and, by (1),
\( \gamma = \varphi(s_2 - m, t_2 + m + 1; X) \neq 0 \) and hence that the monomial \( \gamma x_i^{s_2-m-1}x_j^{t_2+m+1}x_u \) is in this polynomial, i.e.,

\[
\begin{align*}
  s_1 = s_2 & \implies \gamma x_i^{s_1-(m+1)}x_j^{t_1+m+1}x_u \text{ belongs to } p_r \\
  s_1 = s_2 - 1 & \implies \gamma x_i^{s_1-m}x_j^{t_1+m}x_u \text{ belongs to } p_r
\end{align*}
\]

Therefore, the definition of \( \ell \) implies that \( m + 1 \leq \ell \) in the first case and \( m \leq \ell \) in the second, which clearly contradicts the assumption that \( \ell < m \). Consequently, \( x_v \in L(x_i) \) as claimed. A similar argument rules out the case \( x_v \in C(x_j) \).

\[ \square \]

Let \( NC(7.11) \) denote the class of nc polynomials \( p(X, Y) \) of degree \( d \geq 4 \) such that (7.11)

\[
\begin{align*}
  r \geq 2, & \quad \varphi(r, r) \neq 0, \quad \varphi(2r, 0) = \varphi(0, 2r) = 0 \quad \text{and} \\
  \text{if } s > 0, t > 0 \text{ and } s + t = 2r, & \implies \varphi(s, t; X) \neq 0 \iff \varphi(s, t; Y) \neq 0, \\
  \text{if } s > 0, t > 0 \text{ and } s + t = 2r, & \implies \varphi(X; s, t) \neq 0 \iff \varphi(Y; s, t) \neq 0.
\end{align*}
\]

**Theorem 7.11.** Let \( p_1, \ldots, p_{k^2} \) be a family \( \mathcal{P} \) of polynomials in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) of degree \( d \geq 4 \). Then Lemma 7.10, the SVR Algorithm and either Alg.1 or Alg.2 will yield an nc representation \( p(X, Y) \) with \( p \) in the class \( NC(7.11) \) if and only if the given family admits an nc representation in this class.

**Proof.** Under the given assumptions Lemma 2.8 may be applied to obtain the dyslexic diagonal pairs

\[
\{x_{i_1}, x_{j_1}\}, \ldots, \{x_{i_k}, x_{j_k}\}.
\]

To ease the notation, we assume that \( x_{i_s} \) and \( x_{j_s} \) are in the \( ss \) position for \( s = 1, \ldots, k \) and that one of these pairs is partitioned, i.e., for some fixed choice of \( s, x_{i_s} \in X \) and \( x_{j_s} \in Y \).

The objective is to determine \( L(x_{i_s}) \) and \( L(x_{j_s}) \) for each dyslexic diagonal pair. Then for \( s \) fixed, there must exist at least \( k - 1 \) other dyslexic diagonal variables \( \{x_{i_1}, \ldots, x_{i_{s-1}}, x_{i_{s+1}}, \ldots, x_{i_k}\} \) which have the property that

\[
(7.12) \quad L(x_{i_t}) \cap L(x_{i_s}) \neq \emptyset \quad \text{for } t \neq s.
\]

Equation (7.12) implies that the variables \( \{x_{i_1}, \ldots, x_{i_{s-1}}, x_{i_{s+1}}, \ldots, x_{i_k}\} \) lie on the diagonal of \( X \) with \( x_{i_s} \) and that the union of the \( L(x_{i_s}) \) contains all of the variables in \( X \). Therefore, this process serves to partition the variables between \( X \) and \( Y \). Once this is done, the SVR Algorithm and the final steps in Alg.1 or Alg.2 beginning with Algorithm ParPosX will determine an nc representation for \( \mathcal{P} \).

Let \( p_{st} \) denote the polynomial in the \( st \) position with \( s \neq t \) in the array corresponding to \( p(X, Y) \). Then \( p_{st} \), will be of the form

\[
(7.13) \quad p_{st} = x_{i_s}^{r-1}x_{j_s}^{r-1}(ax_{u} + cx_{v}) + x_{i_t}^{r-1}x_{j_t}^{r-1}(bx_{u} + dx_{u}) + x_{i_s}^{r}x_{j_s}^{r-1}(gx_{u} + ex_{v}) + x_{i_t}^{r}x_{j_t}^{r-1}(hx_{u} + fx_{v}) + \cdots.
\]

Given that \( \varphi(r, r) \neq 0 \), we must have that \( \varphi(r, r; X) \neq 0 \) and \( \varphi(r, r; Y) \neq 0 \). and that one of the following cases must hold:
Remark 7.12. The indicated terms in \( p_{st} \) in the first part of the preceding proof can also be grouped as

\[
p_{st} = x_u(a_r x_i x_j x_j + b x_i x_j + x_i x_j + h x_i x_j + x_i x_j) + \cdots.
\]

7.2.4. Two letter monomials \( \triangleright \)-equivalent to \( x_u x_v \). The next result presents another way of determining an nc polynomial representation for families of polynomials containing two letter monomials of the form \( a x_u x_v \) with \( s \geq 2 \) and \( a \in \mathbb{R} \setminus \{0\} \).

Let \( NC(7.14) \) denote the class of nc polynomials of degree \( d \geq 4 \) such that

\[
\begin{aligned}
\varphi(d-1,1) &\neq \varphi(1,d-1), \\
\varphi(d-1,1) &\neq \varphi(d,0) \quad \text{if} \quad \varphi(d-1,1) \neq 0, \\
\varphi(1,d-1) &\neq \varphi(0,d) \quad \text{if} \quad \varphi(1,d-1) \neq 0, \\
\varphi(Y,d-1,1) &\neq 0, \quad \varphi(d-1,1,Y) \neq 0, \\
\varphi(X,1,d-1) &\neq 0, \quad \varphi(1,d-1,X) \neq 0.
\end{aligned}
\]

(7.14)

Proposition 7.13. Let \( p_1, \ldots, p_{2k^2} \) be a family \( \mathcal{P} \) of polynomials in \( 2k^2 \) commuting variables. Then the SVR algorithm will yield an nc representation of \( p(X,Y) \) of \( \mathcal{P} \) with \( p \) in \( NC(7.14) \) if and only if the given family \( \mathcal{P} \) admits an nc representation in this set.
Suppose first that \( \varphi(d - 1, 1) \neq 0 \). The assumptions in the last two lines of (7.14) guarantee that \( k^2 - k \) polynomials will contain four or more terms \( \rhd \)-equivalent to \( x_u^{d-1}x_v \) with \( x_v \neq x_u \), whereas the assumption that \( \varphi(d - 1, 1) \neq \varphi(1, d - 1) \) guarantees that exactly \( k \) polynomials \( p_1, \ldots, p_k \) will contain either one or two monomials \( \rhd \)-equivalent to \( x_u^{d-1}x_v \) with \( x_v \neq x_u \). This allows us to identify them as diagonal entries and to partition them between \( X \) and \( Y \). Therefore, we apply the SVR Algorithm to \( P \) by differentiating \( d - 1 \) times to obtain the family \( \{ P^{d-1}_X p_1, \ldots, P^{d-1}_X p_k \} \) with nc representation

\[
D^{d-1}_{X,i}p(X,Y) = (d-1)!\varphi(d,0)X + (d-1)!\varphi(d-1,1)Y.
\]

Since \( \varphi(d,0) \neq \varphi(d-1,1) \) and \( \varphi(d-1,1) \neq 0 \) by assumption, we can partition the remaining \( k^2 - 2k \) variables between \( X \) and \( Y \).

The construction of an nc polynomial representation can now be completed by invoking the algorithms ParPosX, PosPol, PosY and Algorithm NcCoef. The remaining case follows similarly. \( \square \)

### 7.3. Summary of Two-letter Algorithms

We now summarize our results based on the analysis of two-letter monomials. The methods of this section bear on \( p(X,Y) \) of the form

\[
P(X,Y) = d_1(XY)^t + d_2(YX)^t + d_3(XY)^tX + d_4(YX)^tY + q(X,Y),
\]

where \( q(X,Y) \) is an nc polynomial containing no multiples of the first four monomials in (7.15). The effectiveness of the procedures is summarized by:

**Theorem 7.14.** Let \( p_1, \ldots, p_k \) be a family \( P \) of polynomials in \( 2k^2 \) commuting variables \( x_1, \ldots, x_{2k^2} \) and let \( Q \) denote the set of nc polynomials \( p(X,Y) \) of degree \( d > 1 \) that satisfy the properties in at least one of the following three lists:

1. For some \( s, t \in \mathbb{N} \), \( s \geq t + 2 \geq 4 \), \( \varphi(s, t) \neq 0 \), and \( \varphi(s, t) \neq \varphi(t, s) \). Additionally, assume that \( p(X,Y) \) satisfies one of the following conditions:
   1. \( t\varphi(s, t) \neq (s + 1)\varphi(s + 1, t - 1) \),
   2. \( s\varphi(s, t) \neq (t + 1)\varphi(s - 1, t + 1) \),
   3. \( \varphi(s, t) \neq (s + 1, t - 1; X) \) or \( \varphi(s, t) \neq \varphi(X; s + 1, t - 1) \),
   4. \( \varphi(s, t) \neq \varphi(s - 1, t + 1; Y) \) or \( \varphi(s, t) \neq \varphi(Y; s - 1, t + 1) \).

2. For some \( t \geq 2 \), \( \varphi(t + 1, t) \neq 0 \) and additionally assume that \( p(X,Y) \) satisfies the following properties:
   1. \( \varphi(t + 1, t) \neq d_3 \)
   2. \( \varphi(t + 1, t) \neq d_4 \)
   3. \( \varphi(t + 1, t) \neq \varphi(t, t + 1) \)

3. For some \( r \geq 2 \), \( \varphi(r, r) \neq 0 \) and additionally assume that the nc representation also satisfies the following properties:
   1. \( \varphi(0, 2r) = \varphi(2r, 0) = 0 \)
   2. \( \varphi(r, r) \neq d_1 \)
   3. \( \varphi(r, r) \neq d_2 \)
   4. \( \varphi(s, t; X) \neq 0 \iff \varphi(s, t; Y) \neq 0 \) for all \( s, t \) such that \( s > 0, t > 0, s + t = 2r \).
(5) $\varphi(X; s, t) \neq 0 \iff \varphi(Y; s, t) \neq 0$ for all $s, t$ such that $s > 0, t > 0, s + t = 2r$.

Then the two letter algorithms developed in Section 7.2 determine an nc representation $p(X, Y)$ for $P$ in $Q$ if and only if $P$ has a representation in the class $Q$.

**Proof.** Conditions (1), (2) and (3) are exactly what was needed to make the algorithms described §7.2.1, §7.2.2, and §7.2.3 effective. □

We can now supply a proof that our algorithms work for a collection of families of commutative polynomials that is in direct correspondence with a generic subset of the space of nc polynomials.

7.3.1. **Proof of Theorem 1.2.** Suppose that $\mathcal{U}$ is a subspace of the space $\mathcal{W}$ of nc polynomials of degree $d \geq 4$. Then $\mathcal{U}$ must contain an nc monomial of one of the following forms as a basis element:

$$m_{\alpha, \beta}(X, Y) \quad \text{with} \quad \alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \quad \text{and} \quad |\alpha| + |\beta| = d,$$

where $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ are positive integers, except that $\alpha_1$ and $\beta_n$ are permitted to be equal to zero. Additionally, $\alpha$ and $\beta$ must satisfy one of the following conditions:

1. $|\alpha| = d$ and $|\beta| = 0$ or $|\alpha| = 0$ and $|\beta| = d$,
2. $|\alpha| > 0, |\beta| > 0$ and either $|\alpha| > |\beta| + 1$, or $|\beta| \geq |\alpha| + 1$,
3. $|\alpha| > 0, |\beta| > 0$ and $|\alpha| = |\beta|$,
4. $|\alpha| > 0, |\beta| > 0$ and either $|\alpha| = 1$ or $|\beta| = 1$.

If case (1) occurs, and $p \in \mathcal{U}$ is a polynomial only in the single variable $X$ or $Y$, then the single letter algorithms will determine $p$ and the set

$$S_1 = (NC(4.25) \cup NC(4.27)) \cap \mathcal{U}$$

will be open and dense in $\mathcal{U}$ since the indicated constraints are inequalities. Similarly, in case (2) the set

$$S_2 = (NC(7.7) \cup NC(7.10)) \cap \mathcal{U}$$

will be dense in $\mathcal{U}$

If case (3) occurs and $\mathcal{U}$ contains either the term $X^d$ or $Y^d$ as a basis element, then case (1) applies. If $\mathcal{U}$ does not contain this term as a basis element, then the set

$$S_3 = NC(7.11) \cap \mathcal{U}$$

is open and dense in $\mathcal{U}$.

If case (4) occurs, then the set

$$S_4 = NC(7.14) \cap \mathcal{U}$$

is an open dense set in $\mathcal{U}$ given that the defining constraints are inequality constraints. □
7.4. **Uniqueness results for two-letter algorithms.** The family $\mathcal{Q}$ defined in Theorem 7.14 provides us with a large collection of nc polynomials for which our two letter algorithms will be successful. We now investigate the uniqueness properties of families with a representation in $\mathcal{Q}$.

**Theorem 7.15.** Suppose that $\mathcal{P}$ is a family of $k^2$ polynomials in $2k^2$ commuting variables that admits two nc representations $p(X,Y)$ and $\tilde{p}(X,\tilde{Y})$ in the family $\mathcal{Q}$. Then the matrices $X,Y$ and $\tilde{X},\tilde{Y}$ are permutation equivalent (as defined in (1.13)).

**Proof.** Suppose that (1) of (1.13)) is satisfied and let $\text{diag}\{X\}$ denote the diagonal entries of the matrix $X$. The assumption that both $p$ and $\tilde{p}$ are in $\mathcal{Q}$ implies that either

(a) $\text{diag}\{X\} = \text{diag}\{\tilde{X}\}$ and $\text{diag}\{Y\} = \text{diag}\{\tilde{Y}\}$ or

(b) $\text{diag}\{X\} = \text{diag}\{\tilde{Y}\}$ and $\text{diag}\{Y\} = \text{diag}\{\tilde{X}\}$.

In case (a), if $n \varphi(s,n) \neq (s+1) \varphi(s+1,n-1)$, then Remark 7.3 implies that $D^s_{Y,I_k} p(X,Y)$ (resp., $D^s_{Y,I_k} \tilde{p}(X,\tilde{Y})$) is an nc representation of the family $T_Y p_1, \ldots, T_Y p_{k^2}$ (resp., $T^s_{Y,T_1} p_1, \ldots, T^s_{Y,T_1} p_{k^2}$). Furthermore, given that case (a) holds, the diagonals of $Y$ and $\tilde{Y}$ are the same, which implies that the families $T^s_{Y} p_1, \ldots, T^s_{Y} p_{k^2}$ and $T^s_{Y,T_1} p_1, \ldots, T^s_{Y,T_1} p_{k^2}$ are the same.

Therefore $D^s_{Y,I_k} p(X,Y)$ and $D^s_{Y,I_k} \tilde{p}(X,\tilde{Y})$ are both nc representations of the family $T_Y p_1, \ldots, T_Y p_{k^2}$. Moreover, given that $p, \tilde{p} \in \mathcal{Q}$, the nc polynomials $D^s_{Y,I_k} p(X,Y)$ and $D^s_{Y,I_k} \tilde{p}(X,\tilde{Y})$ are both in the set $\mathcal{W}$ that is defined in Theorem 1.4. Therefore, we may apply Theorem 1.5 to conclude that there exists a permutation matrix $\Pi$ such that either $X = \Pi^T \tilde{X} \Pi$ and $Y = \Pi^T \tilde{Y} \Pi$ or $X = \Pi^T \tilde{X}^T \Pi$ and $Y = \Pi^T \tilde{Y}^T \Pi$.

In case (b), $X = \Pi^T \tilde{Y} \Pi$ and $Y = \Pi^T \tilde{X} \Pi$ or $X = \Pi^T \tilde{Y}^T \Pi$ and $Y = \Pi^T \tilde{X}^T \Pi$. The other three cases in (1.13) are handled in much the same way. \qed

**REFERENCES**


