Separating ABPs and Some Structured Formulas in the Non-Commutative Setting

Prerona Chatterjee
Tata Institute of Fundamental Research, Mumbai, India

Abstract

The motivating question for this work is a long standing open problem, posed by Nisan [20], regarding the relative powers of algebraic branching programs (ABPs) and formulas in the non-commutative setting. Even though the general question remains open, we make some progress towards its resolution. To that effect, we generalise the notion of ordered polynomials in the non-commutative setting (defined by Hrubeš, Wigderson and Yehudayoff [11]) to define abecedarian polynomials and models that naturally compute them.

Our main contribution is a possible new approach towards resolving the $\mathsf{VF}_{\mathrm{nc}}$ vs $\mathsf{VBP}_{\mathrm{nc}}$ question, via lower bounds against abecedarian formulas. In particular, we show the following.

There is an explicit n^2-variate degree d abecedarian polynomial $f_{n,d}(\mathbf{x})$ such that
- $f_{n,d}(\mathbf{x})$ can be computed by an abecedarian ABP of size $O(nd)$;
- any abecedarian formula computing $f_{n,\log n}(\mathbf{x})$ must have size at least $n^{\Omega(\log\log n)}$.

We also show that a super-polynomial lower bound against abecedarian formulas for $f_{\log n,n}(\mathbf{x})$ would separate the powers of formulas and ABPs in the non-commutative setting.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Non-Commutative Formulas, Lower Bound, Separating ABPs and Formulas

Digital Object Identifier 10.4230/LIPIcs.CCC.2021.7

Funding Research supported by the Department of Atomic Energy, Government of India, under project number RTI4001.

Acknowledgements We are thankful to Ramprasad Saptharishi, Mrinal Kumar, C. Ramya and especially Anamay Tengse for the discussions at various stages of this work. We would also like to thank Ramprasad Saptharishi, Anamay Tengse and Kshitij Gajjar for helping with the presentation of the paper. Finally, we would like to thank the anonymous reviewers for their valuable comments that have helped in improving the paper.

1 Introduction

Algebraic Circuit Complexity is the study of multivariate polynomials and their classification based on how hard it is to compute them, using various computational models. The most well studied model is that of algebraic circuits. These are directed acyclic graphs that use algebraic operations like addition and multiplication over some field or ring, to compute polynomials. When the underlying graph is only allowed to be a tree, the model is that of algebraic formulas. The central question in this area is whether the class VNP (algebraic analogue of the class NP) is contained in the class VP (algebraic analogue of the class P). Valiant [23] has shown that the permanent polynomial is complete for VNP, and therefore the VP vs VNP question essentially boils down to asking whether the $n \times n$ permanent can be computed by a $\mathsf{poly}(n)$-sized algebraic circuit.

In this paper, we are interested in polynomials that come from the non-commutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_n\rangle$, where the indeterminates do not commute with each other (that is, $xy \neq yx$ for indeterminates x, y). As a consequence, any monomial in a non-commutative

polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ is essentially a string over the alphabet $\{x_1, \ldots, x_n\}$. This is a natural restriction and there has been a long line of work that studies non-commutative computation beginning with the seminal work of Nisan [20][1].

It was shown by Hrubeš, Wigderson and Yehudayoff [10] that the non-commutative permanent polynomial is complete for the class $\mathsf{VNP}_{\mathrm{nc}}$ (the non-commutative version of VNP). Later Arvind, Joglekar and Raja [1] gave a natural polynomial that is complete for the class of n-variate non-commutative polynomials computable by $\mathsf{poly}(n)$-sized circuits (denoted by $\mathsf{VP}_{\mathrm{nc}}$). The question of whether the classes $\mathsf{VP}_{\mathrm{nc}}$ and $\mathsf{VNP}_{\mathrm{nc}}$ are different is the central open problem in the non-commuatative setting. Although the general question of showing lower bounds against non-commutative circuits remains open, there has been significant progress in restricted settings [17, 16, 15, 22, 8].

With respect to the general question, Hrubeš, Wigderson and Yehudayoff [11] showed that a sufficiently strong super-linear lower bound for the classical sum-of-squares problem implies a separation between $\mathsf{VP}_{\mathrm{nc}}$ and $\mathsf{VNP}_{\mathrm{nc}}$. In another related work, Carmosino, Impagliazzo, Lovett and Mihajlin [5] showed that proving mildly super-linear lower bounds against non-commutative circuits would imply exponential lower bounds against the same model.

One motivation for studying non-commutative computation is that it is possibly easier to prove strong lower bounds in this setting as compared to the usual commutative setting. At least intuitively, it seems harder to *cancel* monomials once they have been calculated when commutativity is not allowed amongst the variables.

For example, the $n \times n$ determinant can be computed by an $O(n^3)$ algebraic circuit, but to the best of our knowledge there is no circuit for the non-commutative determinant of size $2^{o(n)}$. In fact, it was shown by Arvind and Srinivasan [2] that if the non-commutative determinant had a poly-sized circuit, then $\mathsf{VP}_{\mathrm{nc}} = \mathsf{VNP}_{\mathrm{nc}}$.

Even though a super-polynomial lower bound is not known for the non-commutative determinant against circuits, Nisan [20] gave an exponential lower bound on the number of gates in any formula computing it. In contrast, the best lower bound known against formulas in the commutative setting is quadratic[2] [19, 14, 6].

A point to note about the lower bound given by Nisan however, is that the proof actually works for a computational model, called Algebraic Branching Programs (or ABPs), that is believed to be more general than algebraic formulas. In fact, Nisan [20] gave an exact characterisation for the size of any ABP computing a non-commutative polynomial. As far as we are aware, any lower bound known against general non-commutative formulas uses this characterisation and hence is essentially a lower bound against non-commutative ABPs itself.

The motivating question for this work is whether there is a separation between the powers of ABPs and formulas in the non-commutative setting. Let us denote the class of non-commutative polynomials over n variables that can be computed by $\mathsf{poly}(n)$-sized ABPs by $\mathsf{VBP}_{\mathrm{nc}}$. Similarly, let $\mathsf{VF}_{\mathrm{nc}}$ denote the class of non-commutative polynomials over n variables that can be computed by $\mathsf{poly}(n)$-sized formulas. The question is essentially whether $\mathsf{VBP}_{\mathrm{nc}}$ is contained in $\mathsf{VF}_{\mathrm{nc}}$ or not.

This question had been posed by Nisan [20], and the only work we are aware of that has made some progress with respect to this question is the one by Lagarde, Limaye and Srinivasan [15]. They show that certain syntactically restricted non-commutative formulas (called Unique Parse Tree formulas) cannot compute $\mathrm{IMM}_{n,n}$ unless they have size $n^{\Omega(\log n)}$.

In this paper, we study restrictions of a different kind. From here on, we will only be talking about non-commutative computation unless specifically mentioned otherwise.

---

[1] Hyafil [13] had considered non-commutative computation before this, but the main result in that paper is unfortuantely false as shown in [20].

[2] For the elementary symmetric polynomial.

1.1 Abecedarian Polynomials and Models That Compute Them

In [11], Hrubeš et al. have defined the notion of *ordered* polynomials. A homogeneous polynomial of degree d is said to be ordered if the set of variables it depends on can be partitioned into d buckets such that variables occuring in position k only come from the k-th bucket. We generalise this notion by making the bucket indices *position independent*. That is, a variables in position k need not necessarily come from the k-th bucket as long as the variables appear in non-decreasing order of their bucket indices. We call such polynomials abecedarian since, in English, an abecedarian word is one in which all of the letters are arranged in alphabetical order [18].

The difference between ordered polynomials and abecedarian ones can be explained succintly using the notion of *regular expressions* from Automata Theory. For a non-commutative polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$, suppose the variables can been partitioned into buckets $\{X_1, \ldots, X_m\}$. f is said to be *ordered* with respect to $\{X_1, \ldots, X_m\}$ if every monomial in it is a word that can be generated using the *regular expression* $X_1 \cdots X_m$. Note that this is equivalent to set-multilinear polynomials in the commutative setting. On the other hand, f is abecedarian if the monomials in it are words that can be generated using the regular expression $X_1^* \cdots X_m^*$. Subsection 2.1 has a formal definition.

"Getting our Hands Dirty" with Abecedarian Polynomials

Before moving ahead, let us take a look at an example of an abecedarian polynomial. Given a commutative polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, define its non-commutative analogue, $f^{(\mathrm{nc})}$ as follows.

> f and $f^{(\mathrm{nc})}$ look essentially the same, except that variables in every monomial in $f^{(\mathrm{nc})}$ are arranged in non-decreasing order of their indices.

Then, $f^{(\mathrm{nc})}$ is abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$.

Let us also look at a possibly important polynomial that is *not* abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$. Consider the *arc-full rank polynomial*, f, which was constructed by Dvir, Malod, Perifel and Yehudayoff [7] to give a super-polynomial separation between the powers of formulas and ABPs in the multilinear setting.

We look at f as a non-commutative polynomial, f', in the following sense.

> Let $\mathcal{A}$ be the ABP that computes f and think of $\mathcal{A}$ as a non-commutative ABP $\mathcal{A}'$. Then, f' is the polynomial computed by $\mathcal{A}'$.

It is not hard to see that across different monomials in f', the order in which variables are arranged is not consistent. Thus, f' is not abecedarian with respect to the given partition.

A final point to note before we move ahead is that a polynomial might be abecedarian with respect to different partitions[3]. In fact, even the sizes of the different partitions might be different. For example, the polynomial

$$\mathrm{ESYM}_{n,d}^{(\mathsf{ord})} = \sum_{1 \le i_1 < \ldots < i_d \le n} x_{i_1}^{(1)} \cdots x_{i_d}^{(d)}$$

is abecedarian with respect to the partition $\left\{X_k = \left\{x_i^{(k)} \ : \ i \in [n]\right\}\right\}$ which has size d, as well as $\left\{X_i = \left\{x_i^{(k)} \ : \ k \in [d]\right\}\right\}$ which has size n.

[3] Every polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$ is abecedarian with respect to the partition $\{X\}$ for $X = \{x_1, \ldots, x_n\}$.

Abecedarian Models of Computation

Hrubeš et al. [11] have defined *ordered circuits*, a model naturally suited to compute ordered polynomials. We generalise this notion to define circuits that naturally compute abecedarian polynomials. We also define abecedarian ABPs and abecedarian formulas similarly.

Suppose f is an abecedarian polynomial with respect to the partition $\{X_1, \ldots, X_m\}$. For any $1 \leq a \leq b \leq m+1$, $f[a,b)$ is a sub-polynomial of f defined as follows.

- For any $1 \leq a \leq [m+1]$, $f[a,a)$ is the constant term in f.
- For $1 \leq a < b \leq m+1$, $f[a,b)$ contains only those monomials of f in which the first variable is from bucket X_a and the last variable is from any of the buckets in the set $\{X_a, \ldots, X_{b-1}\}$.

A circuit is said to be abecedarian if every gate v in it can be labelled by a tuple (a,b) such that if f_v is the polynomial computed at that gate, then $f_v = f_v[a,b)$. We call a formula abecedarian if it has a similar syntactic property at every gate. For formal definitions, see Definition 15 and Definition 17 respectively. On the other hand, an ABP is said to be abecedarian when every vertex in it can be labelled by a bucket index such that if f is the polynomial computed between vertices labelled with indices a and b respectively, then $f = f[a, b+1)$. Definition 16 is a formal definition.

1.2 Our Main Results

Our main result is a super-polynomial separation between abecedarian formulas and ABPs.

▶ **Theorem 1** (Separating Abecedarian Formulas and Abecedarian ABPs). *Define*

$$\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \leq i_1 \leq \ldots \leq i_d \leq n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

to be the linked *complete homogeneous polynomial over n-variables of degree d. This polynomial is* abecedarian *with respect to the partition $\{X_i \ : \ i \in [n]\}$ if $X_i = \{x_{i,j} \ : \ i \leq j \leq n\}$.*

With respect to this partition,

1. $\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x})$ *has an* abecedarian *ABP of size $O(nd)$;*
2. *any* abecedarian *formula computing* $\mathsf{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ *has size* $n^{\Omega(\log\log n)}$.

That is, there is a super-polynomial separation between abecedarian *formulas and ABPs.*

Our second main result shows that in certain settings, formulas computing abecedarian polynomials can be assumed to be abecedarian without loss of generality.

▶ **Theorem 2** (Converting Formulas into Abecedarian Formulas). *Let f be an* abecedarian *polynomial with respect to a partition of size m, and $\mathcal{F}$ be a formula of size s computing f. If $m = O(\log s)$, then there is an* abecedarian *formula $\mathcal{F}'$ computing f of size* $\mathsf{poly}(s)$.

In other words, an $n^{\omega(1)}$ lower bound against abecedarian formulas computing any polynomial that is abecedarian with respect to a partition of size $O(\log n)$, would result in a super-polynomial lower bound against general non-commutative formulas. These statements suggest a new approach towards resolving the general $\mathsf{VF}_{\mathrm{nc}}$ vs $\mathsf{VBP}_{\mathrm{nc}}$ question.

Connections to the General $\mathsf{VF}_{\mathrm{nc}}$ vs $\mathsf{VBP}_{\mathrm{nc}}$ Question

Theorem 1 gives a separation between abecedarian formulas and ABPs. On the other hand, Theorem 2 shows that if we are given a formula that computes a polynomial that is abecedarian with respect to a partition of *small* size, then we can assume that the formula is abecedarian

without loss of generality. Unfortunately, the partition with respect to which our *hard polynomial* from Theorem 1 is abecedarian, is *not small* in size. Thus, the general question of whether $\mathsf{VBP}_{\mathrm{nc}}$ is contained in $\mathsf{VF}_{\mathrm{nc}}$ or not still remains open. However, there are two natural questions that arise at this point.

1. Can any formula computing an abecedarian polynomial be converted to an abecedarian formula without much blow-up in size, irrespective of the size of the partition?
2. Is there a polynomial f which is abecedarian with respect to a partition that has *small* size such that f witnesses a separation between abecedarian formulas and ABPs?

Clearly, a positive answer to either of these questions would imply that $\mathsf{VBP}_{\mathrm{nc}} \neq \mathsf{VF}_{\mathrm{nc}}$. In particular, a super-polynomial lower bound against abecedarian formulas for a polynomial very similar to the one we used to show our separation would separate $\mathsf{VBP}_{\mathrm{nc}}$ and $\mathsf{VF}_{\mathrm{nc}}$.

▶ **Corollary 3.** *Let the polynomial* linked_CHSYM$_{n,d}(\mathbf{x})$ *be as defined in Theorem 1. An* $n^{\omega(1)}$ *lower bound against* abecedarian *formulas for* linked_CHSYM$_{\log n,n}(\mathbf{x})$ *would imply a super-polynomial separation between non-commutative ABPs and formulas.*

In fact our proof technique also shows that a super-polynomial lower bound against *homogeneous* formulas for our hard polynomial would separate $\mathsf{VBP}_{\mathrm{nc}}$ and $\mathsf{VF}_{\mathrm{nc}}$.

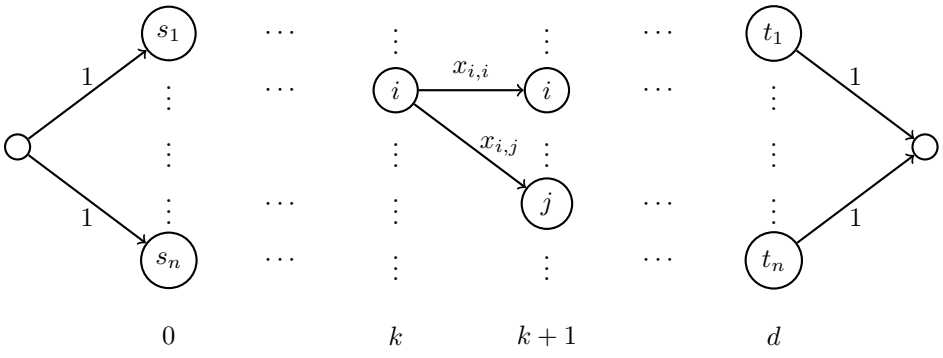
▶ **Corollary 4.** *Let* linked_CHSYM$_{n,d}(\mathbf{x})$ *be as defined in Theorem 1. An* $n^{\omega(1)}$ *lower bound against homogeneous formulas for* linked_CHSYM$_{n,\log n}(\mathbf{x})$ *would result in a super-polynomial separation between ABPs and formulas in the non-commutative setting.*

1.3 Proof Overview

We now give a proof overview of our main theorems.

Separating Abecedarian Formulas and ABPs

Let us first consider Theorem 1.
A *small* abecedarian ABP computing linked_CHSYM$_{n,d}(\mathbf{x})$ is essentially the following.



For the lower bound, assume that we have been given a *small* abecedarian formula computing the polynomial. We then keep modifying this formula till we get a *small* homogeneous multilinear formula computing the *elementary symmetric polynomial* of degree $n/2$. We then use the known lower bound against homogeneous multilinear formulas for this polynomial (shown by Hrubeš and Yehudayoff [12]), to get a contradiction.

Let us spell out the proof in some more detail.

Step 1: Suppose we are given an abecedarian formula computing linked_$\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{\epsilon \log\log n})$. Since the degree of the polynomial being computed is *small*, we can assume that there is in fact a *homogeneous* abecedarian formula computing linked_$\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{c\cdot\epsilon \log\log n})$ for some constant c independent of ϵ.

Step 2: Using the homogeneous abecedarian formula from Step 1, we obtain a more *structured* homogeneous abecedarian formula, of size $O(n^{c\cdot\epsilon \log\log n})$, that computes the same polynomial.

Step 3: We consider the complete homogeneous polynomial over n variables of degree d

$$\mathrm{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1\le i_1\le\ldots\le i_d\le n} x_{i_1}\cdots x_{i_d},$$

and show that there is a homogeneous abecedarian formula of size poly(n) that computes $\mathrm{CHSYM}_{n/2,\log n}(\mathbf{x})$.

Step 4: If the formula in Step 2 has size s and that in Step 3 has size s', then we show that there is a homogeneous abecedarian formula of size $(s\cdot s')$ computing $\mathrm{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$.

Step 5: Next, we show that Step 4 can be used repeatedly at most $O(\log n/\log\log n)$ times, to obtain a homogeneous abecedarian formula computing $\mathrm{CHSYM}_{n/2,n/2}(\mathbf{x})$ of size $O(n^{c\cdot\epsilon \log n})$.

Step 6: Using the formula obtained in Step 5, we get a homogeneous multilinear formula computing the elementary symmetric polynomial of degree $n/2$, of size $O(n^{c\cdot\epsilon \log n})$.

Step 7: Finally, we choose ϵ in such a way that Step 6 contradicts the theorem in [12].

The crucial observation that makes this proof work, is that the polynomial we are working with is structured enough for us to be able to amplify its degree in a systematic way (without blowing up the size by much). This is the 4^{th} step in the description above.

Apart from that, the entire proof essentially boils down to the fact that when formulas are computing low degree polynomials, there are some additional tricks available to make them more structured. A complete proof of Theorem 1 can be found in Section 5.

We now elaborate a little on the first step, since the observations made to prove this step are quite general and possibly useful in various settings. These statements are known to be true in the commutative setting and their proofs in the non-commutative setting are fairly similar to the ones for their commutative counterparts. We state them here nevertheless, since to the best of our knowledge, they have not been stated formally before for the non-commutative setting.

Homogenising Abecedarian Formulas computing Low Degree Polynomials

Raz [21] had shown that if there is a formula computing a homogeneous polynomial of *low* degree in the commutative world, then it can be assumed without loss of generality that the formula is homogeneous. We show that this statement is true even in the non-commutative setting.

▶ **Lemma 5** (Homogenising Abecedarian Formulas computing Low Degree Polynomials)**.** *Suppose f is a non-commutative homogeneous polynomial that can be computed by a fan-in 2 formula, $\mathcal{F}$, of size s, and has degree $d = O(\log s)$. Then there is a homogeneous formula $\mathcal{F}'$ computing f, that has size* poly(s) *and whose multiplication gates have fan-in 2. Further, if $\mathcal{F}$ was* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

The only thing that needs to be checked for Raz's proof to work in this setting is whether non-commutative formulas, and in particular abecedarian formulas can be depth-reduced to log-depth. We show that infact they can be.

Depth Reduction for Abecedarian Formulas

Brent [4] had shown that if there is a formula of size s computing a commutative polynomial f, then there is a formula of depth $O(\log s)$ and size $\mathsf{poly}(s)$ that computes the same polynomial. A similar statemnt was shown by Hrubeš and Wigderson [9] in the non-commutative setting[4]. We show that the statement continues to be true for abecedarian formulas. The proof is exactly along the same lines as the one by Brent [4].

▶ **Lemma 6** (Depth Reduction of Abecedarian Formulas). *If there is a fan-in* 2 *formula* $\mathcal{F}$ *of size* s *computing a non-commutative polynomial* f*, then there is a fan-in* 2 *formula* $\mathcal{F}'$ *of size* $\mathsf{poly}(s)$ *and depth* $O(\log(s))$ *computing* f*. Further if* $\mathcal{F}$ *is homogeneous,* $\mathcal{F}'$ *is also homogeneous. Similarly, if* $\mathcal{F}$ *is* abecedarian *with respect to some partition, then* $\mathcal{F}'$ *is also* abecedarian *with respect to the same partition.*

Converting Formulas into Abecedarian Formulas

Next we go over the proof idea of Theorem 2. In order to prove the statement, we first convert the given formula $\mathcal{F}$ into an abecedarian circuit $\mathcal{C}$, and then unravel $\mathcal{C}$ in order to get an abecedarian formula $\mathcal{F}'$ computing the same polynomial.

The first step is fairly straightforward. The proof is along the same lines as that for homogenising circuits, the only difference being that we keep track of bucket indices of the variables on either ends of the monomials being computed, instead of their degrees.

In the second step, we convert $\mathcal{C}$ into a formula $\mathcal{F}'$. In order to do that, we need to recompute vertices every time it is reused. Thus, to give an upper bound on the size of $\mathcal{F}'$, we need to find an upper bound on the number of distinct paths from any vertex in $\mathcal{C}$ to the root. This analysis is done similarly to the one by Raz [21] in his proof of the fact that formulas computing low degree polynomials can be homogenised efficiently. The requirement of the size of the partition being small also arises because of this analysis.

The only additional point that needs to be checked for the proof to go through is that similar to the commutative setting, non-commutative formulas can be depth reduced as well (Lemma 6). A complete proof of Theorem 2 can be found in Subsection 4.3.

1.4 Other Results: A Complete View of the Abecedarian World

We now go over some other results that helps in completing the view of the abecedarian world. As mentioned earlier, Hrubeš et al. [11] had defined ordered circuits, a model naturally suited to compute ordered polynomials. They had then gone on to show that without loss of generality, any circuit computing an ordered polynomial can be assumed to be ordered[5]. We show that even in the abecedarian setting, such a statement is true.

▶ **Observation 7** (Converting Circuits into Abecedarian Circuits). *Let* f *be an* abecedarian *polynomial with respect to a Partition of size* m*, and* $\mathcal{C}$ *be a circuit of size* s *computing* f*. Then there is an* abecedarian *circuit* $\mathcal{C}'$ *computing* f *of size* $O(m^3 s)$*.*

What this implies is that an $n^{\omega(1)}$ lower bound against abecedarian circuits for any explicit polynomial that is abecedarian would result in a super-polynomial lower bound against general non-commutative circuits. We also show that an analogous statement is true even for abecedarian ABPs.

[4] They in fact showed it for rational functions
[5] Theorem 7.1 in [11].

▶ **Observation 8** (Converting ABPs into Abecedarian ABPs)**.** *Suppose f is an* abecedarian *polynomial with respect to a partition of size m. If there is an ABP $\mathcal{A}$ of size s computing it, then there is an* abecedarian *ABP $\mathcal{A}'$ computing it of size $O(ms)$.*

Next, we define the natural classes of abecedarian polynomials. Let abc - $\mathsf{VP}_{\mathrm{nc}}$ denote the class of abecedarian polynomials that can be computed by poly-sized abecedarian circuits. Similarly let abc - $\mathsf{VBP}_{\mathrm{nc}}$ and abc - $\mathsf{VF}_{\mathrm{nc}}$ denote the classes of abecedarian polynomials that can be computed by poly-sized abecedarian ABPs and abecedarian formulas respectively. We first note that the logical inclusions that should hold, do hold.

▶ **Observation 9** (The Usual Inclusions)**.** *Let* abc - VP_{nc}*,* abc - VBP_{nc} *and* abc - VF_{nc} *denote the classes of* abecedarian *polynomials over n variables that can be computed by* poly(n) *sized* abecedarian *circuits,* abecedarian *ABPs and* abecedarian *formulas respectively. Then,*

$$\text{abc - } \mathsf{VF}_{nc} \subseteq \text{abc - } \mathsf{VBP}_{nc} \subseteq \text{abc - } \mathsf{VP}_{nc}.$$

We also observe that if a degree d polynomial has an abecedarian ABP of size s, then it has an abecedarian formula of size $O(s^{\log d})$ via the usual divide-and-conquer algorithm.

▶ **Observation 10** (Converting Abecedarian ABPs into Abecedarian Formulas)**.** *Suppose f is an* abecedarian *polynomial of degree d. If there is an* abecedarian *ABP $\mathcal{A}$ of size s computing it, then there is an* abecedarian *formula $\mathcal{F}$ computing f of size $O(s^{\log d})$.*

What Theorem 1 essentially shows is that the blow-up observed in Observation 10 is tight.

Finally, it is not hard to see that Nisan's proof can be modified to give an exponential separation between abecedarian ABPs and abecedarian circuits.

General Formula Lower Bound from Homogeneous Formula Lower Bound

We end by showing that homogeneous formula lower bounds for the iterated matrix multiplication polynomial would lead to separating $\mathsf{VF}_{\mathrm{nc}}$ and $\mathsf{VBP}_{\mathrm{nc}}$. This is a corollary of Lemma 5.

▶ **Corollary 11.** *An $n^{\omega(1)}$ lower bound against homogeneous formulas computing the n-variate iterated matrix multiplication polynomial of degree* $\log n$*,* $\mathrm{IMM}_{n,\log n}(\mathbf{x})$*, implies a super-polynomial separation between ABPs and formulas in the non-commutative setting.*

To put the requirement of degree being $O(\log n)$ in perspective, note the following.

▶ Remark 12 (Analogous to Remark 5.12 in [15]). The standard divide and conquer approach for computing the iterated matrix multiplication polynomial $\mathrm{IMM}_{n,d}$ yields a (homogeneous) formula of size $n^{O(\log d)}$. It would be quite surprising if this standard algorithm were not optimal in terms of formula size.

Intuitively, improving on the standard divide and conquer algorithm gets harder as d gets smaller. This is because any (homogeneous) formula of size $n^{o(\log d)}$ for computing $\mathrm{IMM}_{n,d}$ can be used in a straightforward manner to recursively obtain (homogeneous) formulas for $\mathrm{IMM}_{n,D}$ of size $n^{o(logD)}$ for any $D > d$. The case of smaller d, which seems harder algorithmically, is thus a natural first candidate for lower bounds.

1.5 Structure of the Paper

We begin, in Section 2, with formal definitions for abecedarian polynomials and naturally restricted version of circuits, ABPs and formulas that compute them. Then, in Section 3, we prove some structural statements, namely Lemma 6 and Lemma 5. In Section 4, we prove

Theorem 2 along with Observation 7 and Observation 8. We then prove our main result (Theorem 1), that gives a super-polynomial separation between abecedarian formulas and ABPs, in Section 5. Finally, in Section 6, we prove the remaining statements mentioned.

2 Preliminaries

Let us begin by formally defining abecedarian polynomials and the naturally restricted versions of circuits, ABPs and formulas that compute them. Throughout the write-up, we will use $[n]$ to denote the set $\{1, \ldots, n\}$.

2.1 Abecedarian Polynomials

First, we formally define abecedarian polynomials.

▶ **Definition 13** (Abecedarian Polynomials). *A polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$ of degree d is said to be* abecedarian *with respect to a partition $\{X_1, \ldots, X_m\}$ for $\{x_1, \ldots x_n\}$, if*

$$f = f[\emptyset] + \sum_{k=1}^{d} \left(\sum_{1 \le i_1 \le \cdots \le i_k \le m} f[X_{i_1}, \ldots, X_{i_k}] \right)$$

where $f[\emptyset]$ is the constant term in f, and for any $k \in [d]$, $f[X_{i_1}, \ldots, X_{i_k}]$ is defined as follows. For a polynomial f, $f[X_{i_1}, \ldots, X_{i_k}]$ is the homogeneous polynomial of degree k such that for every monomial α,

$$\mathsf{coeff}_\alpha(f[X_{i_1}, \ldots, X_{i_k}]) = \begin{cases} \mathsf{coeff}_\alpha(f) & \text{if } \alpha = x_{\ell_1} \cdots x_{\ell_k} \text{ with } x_{\ell_j} \in X_{i_j} \text{ for every } j \in [k] \\ 0 & \text{otherwise.} \end{cases}$$

In this case, we say that f is abecedarian *with respect to $\{X_1, \ldots, X_m\}$, a partition of size m.*

Abecedarian polynomials are essentially generalisations of ordered polynomials (defined by Hrubeš, Wigderson and Yehudayoff [11]). A homogeneous polynomial, of degree d, is said to be ordered if the set of variables it depends on can be partitioned into d buckets such that variables occuring in position k only come from the k-th bucket.

It is easy to see that any ordered polynomial is also abecedarian with respect to the same partition. This is because position indices are always increasing. For example, consider the following version of the *complete homogeneous symmetric polynomial*.

$$\mathrm{CHSYM}^{(\mathsf{ord})}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x^{(1)}_{i_1} \cdots x^{(d)}_{i_d}.$$

It is both ordered and abecedarian with respect to the partition $\left\{X_k = \left\{x_i^{(k)} \ : \ i \in [n]\right\}\right\}$.

However, note that there are homogeneous polynomials that are abecedarian but not ordered. The following version of the same polynomial is an example.

$$\mathrm{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x_{i_1} \cdots x_{i_d}$$

is abecedarian with respect to $\{X_i \ : \ X_i = \{x_i\}\}$, but is not ordered.

The reason is that for a polynomial to be ordered, the bucket labels have to essentially be position labels. On the other hand, for a polynomial to be abecedarian with respect to a partition, the bucket labels can be independent of position. For example, note that $\mathrm{CHSYM}^{(\mathsf{ord})}_{n,d}(\mathbf{x})$ is abecedarian with respect to the partition $\left\{X_i = \left\{x_i^{(k)} \ : \ k \in [d]\right\}\right\}$ along with the one mentioned earlier.

We now move on to defining algebraic models that naturally compute such polynomials.

2.2 Abecedarian Models of Computation

Homogeneous formulas have the property that any vertex can be labelled by a tuple of position indices (a, b) such that all the monomials being computed at that vertex occur exactly from position a to position b in the final polynomial that is being computed by it. Hrubeš et al. [11] defined *ordered* circuits to be those circuits that have this property.

A circuit computing a degree d polynomial $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ is said to be ordered, if $\{X_1, \ldots, X_d\}$ forms a partition of $\{x_1, \ldots, x_n\}$ such that

- every gate v in the circuit is labelled by a tuple of position indices (a, b);
- if f_v is the polynomial computed at v, then
  - f_v is homogeneous and has degree $(b - a + 1)$;
  - every monomial in f_v is a product of exactly one variable from each of the buckets $X_a, \ldots, X_b$, multiplied in increasing order of their bucket indices.

We generalise this notion to define circuits that naturally compute abecedarian polynomials. Before we can do that, we need the notion of sub-polynomials of any abecedarian polynomial.

▶ **Definition 14** (Sub-Polynomials of an Abecedarian Polynomial)**.** *Suppose f is an* abecedarian *polynomial with respect to the partition $\{X_1, \ldots, X_m\}$, and has degree d. For any $1 \leq a \leq b \leq m + 1$, $f[a, b)$ is the sub-polynomial of f defined as follows.*

- *For any $a \in [m + 1]$, $f[a, a) = f[\emptyset)$ is the constant term in f.*
- *For any $1 \leq a < b \leq m + 1$,*

$$f[a, b) = \sum_{k=1}^{d} \left(\sum_{\substack{i_1, \ldots, i_k \in [m] \\ a = i_1 \leq \cdots \leq i_k < b}} f[X_{i_1}, \ldots, X_{i_k}] \right)$$

where $f[X_{i_1}, \ldots, X_{i_k}]$ is as defined in Definition 13.

Further, we say that a polynomial f is of type *$[a, b)$ if $f = f[a, b)$.*

Let us now formally define abecedarian circuits.

▶ **Definition 15** (Abecedarian Circuits)**.** *For any $a, b \in \mathbb{N}$, let $[a, b)$ denote a set of the form $I = \{i \ : \ a \leq i < b\}$. As a convention, $[a, a)$ denotes the empty set for every $a \in \mathbb{N}$.*

A multi-output circuit $\mathcal{C}$ is said to be abecedarian *when*

- *every gate v in $\mathcal{C}$ is associated with a set $I_v = [a, b)$;*
- *if f_v is the polynomial computed at v, then $f_v = f[a, b)$;*
- *if $v = v_1 + v_2$, then $I_v = I_{v_1} = I_{v_2}$;*
- *if $v = v_1 \times v_2$ with $I_v = [a, a)$, then $I_{v_1} = I_{v_2} = [a, a)$*
- *if $v = v_1 \times v_2$ with $I_v = [a, b)$ and $a < b$, then one of the following is true*
  - *$I_{v_1} = [a, b)$ and $I_{v_2} = [b, b)$;*
  - *$I_{v_1} = [a, a)$ and $I_{v_2} = [a, b)$;*
  - *there exists $a \leq c < b$ such that $I_{v_1} = [a, c + 1)$ and $I_{v_2} = [c, b)$.*

The polynomial computed by $\mathcal{C}$ is the sum of the polynomials computed at the output gates.

Next, we define abecedarian ABPs and abecedarian formulas as the restricted versions of ABPs and formulas respectively, that naturally compute abecedarian polynomials.

2.3 Abecedarian ABPs and Formulas

Homogeneous ABPs have the property that every vertex in it is labelled by a position index such that, polynomials computed between vertices labelled with indices a and b only contain monomials between positions a and $(b-1)$. We define abecedarian ABPs analogously except that the labels on the vertices are bucket labels instead of position labels. These restricted ABPs naturally compute abecedarian polynomials.

▶ **Definition 16** (Abecedarian ABPs). *A multi-input, multi-output ABP $\mathcal{A}$ is said to be* abecedarian *when*

- *every vertex in it is labelled by a bucket index;*
- *if f is the polynomial computed between vertices labelled with indices a and b respectively, then $f = f[a, b+1)$.*

The polynomial computed by $\mathcal{A}$ is the sum of all the polynomials computed between the various (input, output) gate pairs.

Similarly, we define abecedarian formulas as analogues of homogeneous formulas, with the labels again referring to bucket indices instead of position indices.

▶ **Definition 17** (Abecedarian Formulas). *Let sets of the form $[a, b)$, with $a, b \in \mathbb{N}$, be as defined in Definition 15. Suppose $\mathcal{F}$ is a formula computing a polynomial f that is* abecedarian *with respect to a partition of size m. Then $\mathcal{F}$ is said to be* abecedarian *if $\mathcal{F} = \mathcal{F}_1 + \cdots + \mathcal{F}_m$ for sub-formulas $\mathcal{F}_1, \ldots, \mathcal{F}_{m+1}$, where for every $i \in [m+1]$:*

- *$\mathcal{F}_i$ computes the polynomial $f[i, m+1)$;*
- *every gate v in $\mathcal{F}_i$ is associated with a set $I_v = [a, b)$, and in particular, the root node must be associated with the set $[i, m+1)$*
- *if f_v is the polynomial computed at v, then $f_v = f_v[a, b)$;*
- *if $v = v_1 + v_2$, then $I_v = I_{v_1} = I_{v_2}$;*
- *if $v = v_1 \times v_2$ with $I_v = [a, a)$, then $I_{v_1} = I_{v_2} = [a, a)$*
- *if $v = v_1 \times v_2$ with $I_v = [a, b)$ and $a < b$, then one of the following is true*
  - *$I_{v_1} = [a, b)$ and $I_{v_2} = [b, b)$;*
  - *$I_{v_1} = [a, a)$ and $I_{v_2} = [a, b)$;*
  - *there exists $a \leq c < b$ such that $I_{v_1} = [a, c+1)$ and $I_{v_2} = [c, b)$.*

Further, $\mathcal{F}$ is said to be homogeneous if each $\mathcal{F}_i$ is homogeneous.

With these definitions in mind, we now move to proving some structural statements.

3 Structural Statements

In this section, we prove two structural statements in the non-commutative setting that are known to be true in the commutative setting. Apart from being crucial to our proofs, they are possibly interesting observations in their own right.

3.1 Depth Reduction for Non-Commutative Formulas

Brent [4] had shown that if there is a formula of size s computing a commutative polynomial f, then there is a formula of depth $O(\log s)$ and size $\mathsf{poly}(s)$ that computes the same polynomial. We show that this is also true in the non-commutative setting.

The proof is essentially the same as the one by Brent [4], just analysed carefully. We give the complete proof for the sake of completeness.

▶ **Lemma 6** (Depth Reduction of Abecedarian Formulas). *If there is a fan-in 2 formula $\mathcal{F}$ of size s computing a non-commutative polynomial f, then there is a fan-in 2 formula $\mathcal{F}'$ of size* $\mathsf{poly}(s)$ *and depth $O(\log(s))$ computing f. Further if $\mathcal{F}$ is homogeneous, $\mathcal{F}'$ is also homogeneous. Similarly, if $\mathcal{F}$ is* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

Proof. Suppose $\mathcal{F}$ is a fan-in 2 formula of size s that computes f. Then, we claim the following.

▷ Claim 18. Suppose $\mathcal{F}_0$ is a formula computing a polynomial f_0 and has fan-in 2. Then the there exist sub-formulas, $L, \mathcal{F}_1, R, \mathcal{F}_2$, of $\mathcal{F}_0$ such that

- $\mathcal{F}_0' = L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ also computes f_0;
- each of $L, \mathcal{F}_1, R, \mathcal{F}_2$ have size at least $(s/3)$ and at most $(2s/3)$;
- if $\mathcal{F}_0$ is homogeneous, then so are $L, \mathcal{F}_1, R, \mathcal{F}_2$;
- if $\mathcal{F}_0$ is abecedarian with respect to some partition, f_{left}, f_1, f_{right}, f_2 are polynomials computed by $L, \mathcal{F}_1, R, \mathcal{F}_2$ respectively and $f_0 = f_0[a, b)$, then $f_2 = f_2[a, b)$ and
  - each of $L, \mathcal{F}_1, R, \mathcal{F}_2$ are abecedarian with respect to the same partition as $\mathcal{F}_0$
  - when $a = b$, $\quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, a) \quad f_1 = f_1[a, a) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[a, a)$;
  - when $a < b$, there exist $a \le i \le j \le b$ such that

$$
\begin{aligned}
a = i < j = b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) \quad f_1 = f_1[i, j) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a = i = j < b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) \quad f_1 = f_1[i, j) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a = i < j < b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i) \quad f_1 = f_1[i, j+1) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a < i = j = b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) \quad f_1 = f_1[i, j) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a < i = j < b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) \quad f_1 = f_1[i+1, j+1) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a < i < j = b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) \quad f_1 = f_1[i, j) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).\\
a < i < j < b \quad &\implies \quad f_{\mathsf{left}} = f_{\mathsf{left}}[a, i+1) \quad f_1 = f_1[i, j+1) \quad f_{\mathsf{right}} = f_{\mathsf{right}}[j, b).
\end{aligned}
$$

Before proving Claim 18, let us complete the proof of Lemma 6 using it.

By the above claim, we have a formula $\mathcal{F}_0'$ computing f_0 that looks like $L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ where each of $L, \mathcal{F}_1, R, \mathcal{F}_2$ have size at most $(2s/3)$. Further if $\mathcal{F}$ is homogeneous, then so are each of $L, \mathcal{F}_1, R, \mathcal{F}_2$. Hence, $\mathcal{F}_0'$ is homogeneous. On the other hand, when $\mathcal{F}_0$ is abecedarian, so are $L, \mathcal{F}_1, R, \mathcal{F}_2$. Further, note that $\mathcal{F}_0'$ is also abecedarian in this case since $f_{\mathsf{left}}, f_1, f_{\mathsf{right}}, f_2$ are of the *correct type* due to Claim 18.

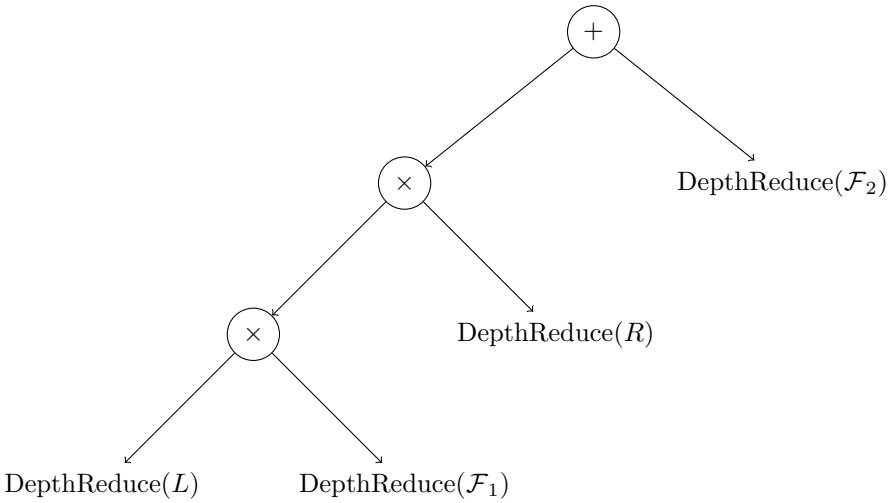
In all the cases, recursively applying this technique, on each of $L, \mathcal{F}_1, R, \mathcal{F}_2$, we get

$$\mathsf{depth}(s) \le \mathsf{depth}(2s/3) + 3 \qquad \text{and} \qquad \mathsf{size}(s) \le 4 \cdot \mathsf{size}(2s/3) + 3.$$

Note that in the base case, when s is constant, both $\mathsf{size}(s)$ and $\mathsf{depth}(s)$ are constants. Thus,

$$\mathsf{depth}(s) = O(\log s) \qquad \text{and} \qquad \mathsf{size}(s) = \mathsf{poly}(s).$$ ◀

Pictorially, once we have Claim 18, we essentially do the following recursively.



We now complete the proof of Claim 18.

Proof of Claim 18. From the root let us traverse $\mathcal{F}_0$ towards the leaves, always choosing the child that has a larger sub-tree under it, till we find a vertex v such that the associated sub-tree has size at most $(2s/3)$. Since $\mathcal{F}_0$ tree has fan-in 2, we also know that the size of this sub-tree must be at least $(s/3)$. Let this sub-tree be $\mathcal{F}_1$. Additionally, in the case when $\mathcal{F}_0$ is abecedarian, let us assume that v is labelled with $[i_v, j_v)$.

Let $\mathcal{P}$ be the path from v to the root and v_{add} the addition gate on $\mathcal{P}$ which is closest to v. Also let the set of multiplication gates on $\mathcal{P}$ be $\{v_1, \ldots, v_\ell\}$ for some $\ell \in \mathbb{N}$. Assume, without loss of generality, that v_1 is closest to v and v_ℓ to the root. Further, for every $i \in [\ell]$, let L_i be sub-formula corresponding to the left child of v_i and R_i the one to its right child. Note that for every $i \in [\ell]$, exactly one of children of v_i is a vertex in $\mathcal{P}$. We can then define L and R as follows.

Step 1: Set $L = R = 1$.

Step 2: For i from 1 to ℓ,

$$L = \begin{cases} L_i \times L & \text{if the right child of } v_i \text{ is a vertex in } \mathcal{P}, \\ L & \text{otherwise.} \end{cases}$$

and

$$R = \begin{cases} R & \text{if the right child of } v_i \text{ is a vertex in } \mathcal{P}, \\ R \times R_i & \text{otherwise.} \end{cases}$$

Also define $\mathcal{F}_2$ to be the formula we get by replacing the vertex v and the sub-tree under it with 0, and then removing the redundant gates.

Clearly, by construction, $\mathcal{F}_1$, L, R and $\mathcal{F}_2$ are sub-formulas of $\mathcal{F}_0$. Further, $\mathcal{F}_1$ is disjoint from L, R and $\mathcal{F}_2$. As a result, since $\mathcal{F}_1$ has size at least $(s/3)$ and at most $(2s/3)$, it must be the case that each of L, R and $\mathcal{F}_2$ have size at least $(s/3)$ and at most $(2s/3)$.

Also, it is not hard to see that $\mathcal{F}_0' = L \cdot \mathcal{F}_1 \cdot R + \mathcal{F}_2$ computes f_0. What is left to check is that when $\mathcal{F}_0$ is homogeneous or abecedarian, then $L, \mathcal{F}_1, R, \mathcal{F}_2$ have the additional properties claimed. The one line proof of this is that each *parse-tree*[6] of $\mathcal{F}_0$ is merely restructured in the above process, without changing its value. We however go over the proof explicitly for the sake of completeness.

[6] For a definition, see for example [15].

When $\mathcal{F}_0$ is homogeneous, since $L, \mathcal{F}_1, R, \mathcal{F}_2$ are sub-formulas, they are also homogeneous. On the other hand, suppose $\mathcal{F}_0$ is abecedarian and $f_0 = f_0[a, b)$. Recall that the vertex v was labelled by $[i_v, j_v)$. Let us set $i = i_v$ and $j = j_v$. Then, by definition, $\mathcal{F}_1$ is labelled by $[i, j)$. Hence, if f_1 is the polynomial computed at v, then $f_1 = f_1[i, j)$. Further, $\mathcal{F}_1$ is abecedarian since it is a sub-formula of $\mathcal{F}_0$ and computes an abecedarian polynomial.

Now let us focus on $\mathcal{F}_2$. Essentially $\mathcal{F}_2$ is got by removing from $\mathcal{F}_0$, v and all the multiplication gates on P between v and v_{add} along with the sub-trees under them. Thus $\mathcal{F}_2$ is also abecedarian in this case, and if f_2 is the polynomial by it, then $f_2 = f_2[a, b)$.

Finally, note that the left indices of labels on the various vertices of $\mathcal{P}$ change only at the gates at which multiplications to L occur. Further, note that they occur in the *correct order* and are of the *correct type*. Thus, by induction, it is easy to see that the labels on L are consistent with those on the L_is when the respective multiplications happen. Therefore L is abecedarian, and $f_{\mathsf{left}} = f_{\mathsf{left}}[a, i)$.

For similar reasons, R is also abecedarian and $f_{\mathsf{right}} = f_{\mathsf{right}}[j, b)$. This completes the proof. ◁

3.2 Homogenisation

Raz [21] had shown that if there is a formula computing a homogeneous polynomial of *low* degree in the commutative world, then it can be assumed without loss of generality that the formula is homogeneous. We show that his proof also works in the non-commutative setting because of Lemma 6. A complete proof is given here for the sake of completeness.

▶ **Lemma 5** (Homogenising Abecedarian Formulas computing Low Degree Polynomials)**.** *Suppose f is a non-commutative homogeneous polynomial that can be computed by a fan-in 2 formula, $\mathcal{F}$, of size s, and has degree $d = O(\log s)$. Then there is a homogeneous formula $\mathcal{F}'$ computing f, that has size* poly(s) *and whose multiplication gates have fan-in 2. Further, if $\mathcal{F}$ was* abecedarian *with respect to some partition, then $\mathcal{F}'$ is also* abecedarian *with respect to the same partition.*

Proof. We first note that since s is the ABP complexity of f, $s' \geq s$. Further if $\mathcal{F}$ has depth r, then by Lemma 6, we can assume without loss of generality, that $r = O(\log s')$.

In order to construct a homogeneous formula computing f, we first homogenise $\mathcal{F}$ to obtain a circuit $\mathcal{C}$, and then *unravel* $\mathcal{C}$ to make it into a formula $\mathcal{F}'$.

The first step is done in the usual manner. For every gate v in $\mathcal{F}$, we have $d + 1$ gates $(v, 0)$, …, (v, d) in $\mathcal{C}$. Intuitively if f_v is the polynomial computed at v, then the polynomial computed at (v, i) is the degree i homogeneous component of f_v. These vertices are then connected as follows.

- If $v = u_1 + u_2$, then for every $i \in \{0, \ldots, d\}$, $\quad (v, i) = (u_1, i) + (u_2, i)$.
- If $v = u_1 \times u_2$, then for every $i \in \{0, \ldots, d\}$, $\quad (v, i) = \sum_{j=0}^{i} (u_1, j) \times (u_2, i - j)$.

So, we now have a homogeneous circuit $\mathcal{C}$ that computes f and has size at most $O(d^2 \cdot s')$. Also, the depth of this circuit is at most twice that of $\mathcal{F}$, and the multiplication gates have fan-in 2. To convert $\mathcal{C}$ into a formula $\mathcal{F}'$, we have to recompute nodes whenever they have to be reused. That is, a particular vertex in $\mathcal{C}$ has to be duplicated as many times as there are paths from the vertex to the root. Thus, to upper bound the size of $\mathcal{F}'$, we need to give an upper bound on the number of distinct paths from every vertex of $\mathcal{C}$ to its root.

Let us arbitrarily choose a vertex (v, i) in $\mathcal{C}$, and consider the path from it to the root. Suppose the path is $(v, i) = (v_1, i_1) \to \cdots \to (v_\ell, i_\ell) = (\mathsf{root}, d)$ where ℓ is at most the depth of $\mathcal{C}$. Now since $\mathcal{C}$ comes from a formula, the only reason multiple paths can exist is because

of the second index, and therefore it is enough to focus on that. Note that it must be the case that $i = i_1 \leq \cdots \leq i_\ell = d$. Hence, if we define $\delta_j = i_{j+1} - i_j$ for $j \in [\ell - 1]$, then the δ_js are non-negative integers such that $\delta_1 + \cdots + \delta_{\ell-1} = (d - i)$. Thus, the number of choices we have for $(i_2, \ldots, i_\ell)$ such that $i = i_1 \leq \cdots \leq i_\ell = d$, is the same as the number of choices we have for $(\delta_1, \ldots, \delta_{\ell-1})$ such that $\delta_1 + \cdots + \delta_{\ell-1} = (d - i) \leq d$. This is at most $\binom{\ell+d}{\ell}$. Note that in this process the fan-in of the gates have not changed, and hence the multiplication gates in $\mathcal{F}'$ continue to have fan-in 2. Further, we know that the $\mathcal{C}$ has depth $2r$ and hence $\ell \leq 2r$. Therefore, the number of paths from (v, i) to the root is at most $\binom{2r+d}{2r}$. Hence, if $\mathcal{F}'$ is the formula obtained by unravelling $\mathcal{C}$, then $\mathsf{size}(\mathcal{F}') \leq s' \cdot d^2 \cdot \binom{2r+d}{d}$. Here $r = O(\log(s'))$, and $s \leq s'$ implying that $d = O(\log(s)) = O(\log(s'))$. Thus, $\mathsf{size}(\mathcal{F}') \leq \mathsf{poly}(s')$.

Finally, assume that $\mathcal{F}$ is abecedarian. Then every vertex v is labelled with a tuple of bucket indices, say (a_v, b_v). In that case, we add the label (a_v, b_v) to the gates $\{(v, i)\}_{i=0}^{d}$ in $\mathcal{C}$ and continue with the proof as is. Note that the final formula that we get, $\mathcal{F}'$, is abecedarian and all the other properties that were true in the general case, continue to be true. ◀

4 Converting Computational Models into Abecedarian Ones

In this section we show that, without loss of generality, circuits and ABPs computing abecedarian polynomials can be assumed to be abecedarian. For formulas however, we can prove such a statement only in certain cases.

4.1 Circuits

Hrubeš et al. [11] had shown that any circuit computing an ordered polynomial can be assumed to be ordered without loss of generality.

▶ **Theorem 19** (Theorem 7.1 in [11]). *Let $\mathcal{C}$ be a circuit of size s computing an ordered polynomial f of degree d. Then, there is an ordered circuit $\mathcal{C}'$ of size $O(d^3 s)$ that computes f.*

We show that the proof of this statement can be generalised to show Observation 7. A complete proof is given for the sake of completeness.

▶ **Observation 7** (Converting Circuits into Abecedarian Circuits). *Let f be an* abecedarian *polynomial with respect to a Partition of size m, and $\mathcal{C}$ be a circuit of size s computing f. Then there is an* abecedarian *circuit $\mathcal{C}'$ computing f of size $O(m^3 s)$.*

Proof. Without loss of generality, let us assume that $\mathcal{C}$ has fan-in 2.

We prove the given statement by describing how to construct $\mathcal{C}'$ from $\mathcal{C}$. For each gate v in $\mathcal{C}$, we make $O(m^2)$ copies in $\mathcal{C}'$, $\{(v, [a, b)) \ : \ 1 \leq a \leq b \leq m + 1\}$; and if root is the output gate in $\mathcal{C}$, then we define the set of output gates in $\mathcal{C}'$ to be $\{(\mathsf{root}, [i, m + 1))\}_{i \in [m+1]}$.

Intuitively, if f_v is the polynomial computed at v in $\mathcal{C}$, then the polynomial computed at $(v, [a, b))$ is $f_v[a, b)$. Thus if f was the polynomial computed at root, then the polynomial computed by $\mathcal{C}'$ is $\sum_{i=1}^{m+1} f[i, m + 1)$ which is indeed f.

We ensure this property at every gate by adding edges as follows.

- If v is an input gate labelled by a field element γ,
  - we set $(v, [a, a)) = \gamma$ for every $a \in [m + 1]$;
  - we set $(v, [a, b)) = 0$ for every $1 \leq a < b \leq m + 1$.
- If v is an input gate labelled by a variable x_i and $x_i \in X_k$,
  - we set $(v, [k, k + 1)) = x_i$;
  - we set $(v, [a, b)) = 0$ for every $a \neq k$, $b \neq k + 1$.

- If $v = v_1 + v_2$, we set $(v, [a, b)) = (v_1, [a, b)) + (v_2, [a, b))$ for every $a \leq b \in [m + 1]$.
- If $v = v_1 \times v_2$, we set $(v, [a, a)) = (v_1, [a, a)) \cdot (v_2, [a, a))$ for every $a \in [m + 1]$; and

$$(v, [a, b)) = (v_1, [a, a)) \cdot (v_2, [a, b)) + (v_1, [a, b)) \cdot (v_2, [b, b)) + \sum_{c=a}^{b-1} (v_1, [a, c + 1)) \times (v_2, [c, b))$$

for every $1 \leq a < b \leq m + 1$.

Finally, for every $1 \leq a \leq b \leq m + 1$, we associate the gate $(v, [a, b))$ in $\mathcal{C}'$ with the set $[a, b)$.

Using induction, one can easily show that the gates in $\mathcal{C}'$ have the claimed properties. Hence $\mathcal{C}'$ is indeed an abecedarian circuit computing f. Further for every gate v in $\mathcal{C}$, there are at most $O(m^3)$ vertices in $\mathcal{C}'$. Thus the size of $\mathcal{C}'$ is $O(m^3 s)$. ◀

4.2 Algebraic Branching Programs

Next, we show that a similar statement is true for ABPs as well.

▶ **Observation 8** (Converting ABPs into Abecedarian ABPs)**.** *Suppose f is an* abecedarian *polynomial with respect to a partition of size m. If there is an ABP $\mathcal{A}$ of size s computing it, then there is an* abecedarian *ABP $\mathcal{A}'$ computing it of size $O(ms)$.*

Proof. Let f have degree d and be abecedarian with respect to the buckets $\{X_i\}_{i=1}^m$, where $X_i = \{x_{i,j} \ : \ j \in [n_i]\}$. Without loss of generality, we can assume that $\mathcal{A}$ is homogeneous[7]. If f is not homogeneous, $\mathcal{A}$ can be thought of as a collection of homogeneous ABPs $\{\mathcal{A}_1, \ldots, \mathcal{A}_d\}$ where $\mathcal{A}_k$ computes the k-th homogeneous component of f.

We prove the theorem by describing how to construct $\mathcal{A}'$. For each vertex v in $\mathcal{A}$, make $O(m)$ copies in $\mathcal{A}'$, namely $\{(v, a) \ : \ 0 \leq a \leq m\}$. Intuitively, if $g_{(u,v)}$ is the polynomial computed between u and v in $\mathcal{A}$, then the polynomial computed between (u, a) and (v, b) in $\mathcal{A}'$ is $g_{(u,v)}[a, b + 1)$. The way we ensure this property at every vertex is by adding edges in $\mathcal{A}'$ as follows.

> For any two vertices u, v in $\mathcal{A}$, suppose there is an edge between them that is labelled with $\sum_{i \in [m]} \sum_{j \in [n_i]} \gamma_{i,j} x_{i,j}$. Then, for every $a, b \in [m]$ with $a \leq b$, add an edge from (u, a) to (v, b) with label $\sum_{i=a}^{b} \left(\sum_{j \in [n_i]} \gamma_{i,j} x_{i,j} \right)$.

Also, associate the bucket index a with the gate (v, a) in $\mathcal{A}'$.

By induction, one can easily show that the gates in $\mathcal{A}'$ have the claimed property. Hence $\mathcal{A}'$ is indeed an abecedarian ABP computing f. Further, every vertex v in $\mathcal{A}$, there are at most $O(m)$ vertices in $\mathcal{A}'$. Therefore, the size of $\mathcal{A}'$ is $O(ms)$. ◀

4.3 Formulas

Finally we show that in the case of formulas, we can prove a similar statement only when the polynomial is abecedarian with respect to a partition of *small size*. The proof is very similar to that of Lemma 5.

▶ **Theorem 2** (Converting Formulas into Abecedarian Formulas)**.** *Let f be an* abecedarian *polynomial with respect to a partition of size m, and $\mathcal{F}$ be a formula of size s computing f. If $m = O(\log s)$, then there is an* abecedarian *formula $\mathcal{F}'$ computing f of size* poly(s).

[7] Every edge is labelled by a homogeneous form.

Proof. Let us assume additionally that $\mathcal{F}$ has depth r. Now Lemma 6 implies that $r = \log(s)$ without loss of generality. By Observation 7, there is an abecedarian circuit $\mathcal{C}$ that computes f and has size at most $s' = O(s \cdot m^3)$. Further its proof implies that the depth of $\mathcal{C}$ is at most $2r$.

To convert $\mathcal{C}$ into an abecedarian formula $\mathcal{F}'$, we have to recompute a node each time it has to be reused. That is, a particular vertex in $\mathcal{C}$ has to be duplicated as many times as there are paths from the vertex to the root. Thus to upper bound the size of $\mathcal{F}'$, we need to give an upper bound on the number of distinct paths from every vertex in $\mathcal{C}$ to its root.

Let us arbitrarily choose a vertex $(v, [a, b))$ in $\mathcal{C}$, and consider the path from it to the root. Suppose the path is $(v, [a, b)) = (v_1, [a_1, b_1)) \to \cdots \to (v_\ell, [a_\ell, b_\ell)) = (\mathsf{root}, [i, m + 1))$ for some ℓ that is at most the depth of $\mathcal{C}$. Note that it must be the case that

$$i \leq a_\ell \leq \cdots \leq a_1 \leq a \leq b \leq b_1 \leq b_\ell \leq m + 1.$$

Let us define $\delta_j = a_j - a_{j+1}$ and $\delta'_j = b_{j+1} - b_j$ for $j \in [\ell - 1]$. Then, the number of choices we have for $(a_1, \ldots, a_\ell)$ and $(b_1, \ldots, b_\ell)$ such that

$$i = a_\ell \leq \cdots a_1 = a \leq b = b_1 \leq \cdots \leq b_\ell = m + 1$$

is the same as the number of choices we have for $(\delta_1, \ldots, \delta_{\ell-1}, \delta'_1, \ldots, \delta'_{\ell-1})$ such that

$$\delta_1 + \cdots + \delta_{\ell-1} + \delta'_1 + \cdots + \delta'_{\ell-1} = (m + 1 - (b - a) - i) \leq m.$$

This is clearly at most $\binom{2\ell+m}{m}$.

Further, we know that the $\mathcal{C}$ has depth $2r$ and hence $\ell \leq 2r$. Therefore, the number of paths from (v, i) to the root is at most $\binom{4r+m}{m}$. Hence if $\mathcal{F}'$ is the formula obtained by unravelling $\mathcal{C}$, then $\mathsf{size}(\mathcal{F}') \leq s' \cdot m^2 \cdot \binom{4r+m}{m}$. Here $s' = O(m^3 \cdot s)$, $r = O(\log(s))$ and $m = O(\log(s))$. Thus, $\mathsf{size}(\mathcal{F}') \leq \mathsf{poly}(s)$. ◀

5 Separating Abecedarian ABPs and Abecedarian Formulas

In this section, we prove our main theorem: a super-polynomial separataion between the powers of abecedarian formulas and ABPs. Before proceeding to the proof however, we first go over some observations that will help us with the proof.

5.1 Some Simple Observations

The two main polynomials we will be working with are $\mathsf{linked\_CHSYM}_{n,d}$ and $\mathrm{CHSYM}_{n,d}$. Let us recall their definitions.

$$\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \leq i_1 \leq \ldots \leq i_d \leq n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right),$$

is abecedarian with respect to the partition $\{X_1, \ldots, X_n\}$ where $X_i = \{x_{i,j} \ : \ j \in [n]\}$, and

$$\mathrm{CHSYM}_{n,d}(\mathbf{x}) = \sum_{1 \leq i_1 \leq \ldots \leq i_d \leq n} x_{i_1} \cdots x_{i_d}.$$

is abecedarian with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$.

We begin with the notion of a *linked* abecedarian formula computing $\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x})$.

▶ **Definition 20.** *An* abecedarian *formula computing* linked_CHSYM$_{n,d}$ *is said to be* linked *if at every gate, all the monomials occuring in the polynomial computed at that gate have the following property.*

x_{ij} *appears right before* $x_{i'j'}$ *in the monomial* $\implies j = i'$.

The first observation shows that any abecedarian formula computing linked_CHSYM$_{n,d}(\mathbf{x})$ can be assumed to be *linked* without loss of generality.

▶ **Observation 21.** *Let* $\mathcal{F}$ *be a homogeneous* abecedarian *formula of size* s *that computes* linked_CHSYM$_{n,d}(\mathbf{x})$*, and let the multiplication gates of* $\mathcal{F}$ *have fan-in* 2*. Then there is a homogeneous* linked abecedarian *formula* $\mathcal{F}'$ *computing the same polynomial of size* $O(s)$*.*

Proof. For any leaf ℓ in $\mathcal{F}$ labelled by a variable, say $x_{i,j}$, suppose $\mathcal{P}$ is the path from ℓ to the root. Consider the set of multiplication gates on $\mathcal{P}$ whose left child is part of $\mathcal{P}$, and let v be the one that is closest to ℓ. Since $\mathcal{F}$ is abecedarian, the right child of v must be associated with a set, say $[a, b)$. If $j \neq a$, we set the label of ℓ to zero; otherwise we let it be $x_{i,j}$.

Note that this operation does not kill any *valid* monomial. Let $\mathcal{F}'$ be the formula we get by performing the above operation on every leaf of $\mathcal{F}$ that is labelled by a variable. $\mathcal{F}'$ is clearly homogeneous and abecedarian. We show that $\mathcal{F}'$ is also *linked*.

Suppose that is not the case. Then there is must be a *problematic* vertex in $\mathcal{F}'$. Let v be such a vertex of minimal height. That is, there is a monomial in the polynomial computed at v in which, say, $x_{i,j}$ appears right before $x_{i',j'}$ but $j \neq i'$. Further, the sub-formulas corresponding to the children of v are linked. Note that v must be a multiplication gate; not a leaf or an addition gate.

Let f_{left} and f_{right} be the polynomials computed at the left and right children of v respectively. Also, let $[a, b)$ be the set associated with the right child of v. Then, it must be the case that the first variable in any monomial in f_{right} looks like $x_{a,j'}$ for some j'. Further, there must be a monomial in f_{left} in which the last variable looks like $x_{i,j}$ for $j \neq a$.

Look at the leaf corresponding to this variable. Let this leaf be ℓ and let $\mathcal{P}$ be the path from ℓ to the root. Since $x_{i,j}$ is the right most variable in f_{left}, it must be the case that v is the multiplication gate that is closest to ℓ, whose left child is on $\mathcal{P}$. But then, we should have set $x_{i,j}$ to zero since $j \neq a$. Hence, such a monomial can not appear in f_{left}.

This shows that $\mathcal{F}'$ is indeed a homogeneous *linked* abecedarian formula of size at most that of $\mathcal{F}$ that computes linked_CHSYM$_{n,d}(\mathbf{x})$. ◀

The next observation shows that there is a poly-sized homogeneous abecedarian formula that computes CHSYM$_{n,\log n}(\mathbf{x})$.

▶ **Observation 22.** CHSYM$_{n/2,\log n}(\mathbf{x})$ *can be computed by a homogeneous* abecedarian *formula of size* poly(n)*.*

Proof. Consider the following polynomial over variables $\{t, x_1, \ldots, x_n\}$, where we think of t as a commuting variable and $x_1, \ldots, x_n$ as non-commuting variables.

$$f_{n,d}(\mathbf{x}) = \prod_{i=1}^{n} \left(1 + \sum_{j=1}^{d} t^j \cdot x_i^j\right)$$

Note that the coefficient of t^d in $f_{n,d}(\mathbf{x})$ is exactly CHSYM$_{n,d}(\mathbf{x})$. Further, it is not hard to see that $f_{n/2,\log n}(\mathbf{x})$ is abecedarian in terms of $\mathbf{x}$ with respect to the partition $\{X_i \ : \ X_i = \{x_i\}\}$, and that the given expression results in an abecedarian formula of size $O(n(\log n)^2)$.

Since t is a commuting variable, we can use the usual interpolation techniques [3], to get an abecedarian formula computing $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n \log n \cdot n(\log n)^2) = O(n^2(\log n)^3) = \text{poly}(n)$. Since the degree of $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ is $O(\log n)$, by Lemma 5, there is a homogeneous abecedarian formula computing $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $\text{poly}(n)$. ◀

Another simple observation is that if we are given a homogeneous abecedarian formula for an abecedarian polynomial, then we almost immediately have one for its various sub-polynomials.

▶ **Observation 23.** *Suppose there is a homogeneous* abecedarian *formula $\mathcal{F}$ computing a polynomial f that is* abecedarian *with respect to a partition of size m. Then, for any $a, b \in [m+1]$, there is a homogeneous* abecedarian *formula $\mathcal{F}_{a,b}$ of size s that computes $f[a, b)$.*

Proof. Recall that if $\mathcal{F}$ is a homogeneous abecedarian formula computing f, then $\mathcal{F}$ is in fact a set of formulas $\{\mathcal{F}_i : \mathcal{F}_i \text{ computes } f[i, m+1)\}$. Consider the formula $\mathcal{F}_a$ and set all variables that belong to buckets $\{X_b, \ldots, X_m\}$ to zero in $\mathcal{F}_a$. This operation clearly kills exactly the monomials in $f[a, m+1)$ that are not in $f[a, b)$. Thus if we call this new formula $\mathcal{F}_{a,b}$, then $\mathcal{F}_{a,b}$ is homogeneous, abecedarian and computes $f[a, b)$. ◀

The next observation is extremely crucial, since it allows us to *amplify the degree* of $\text{CHSYM}_{n,d}$.

▶ **Lemma 24.** *Suppose there is a homogeneous* abecedarian *formula computing $\text{CHSYM}_{n,d}(\mathbf{x})$ of size s, and a homogeneous* linked abecedarian *formula computing $\text{linked\_CHSYM}_{n,d'}(\mathbf{x})$ of size s'. Then, there is a homogeneous* abecedarian *formula computing $\text{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ of size $(s \cdot s')$.*

Proof. Let $\mathcal{F}$ be the homogeneous abecedarian formula computing $\text{CHSYM}_{n,d}(\mathbf{x})$ of size s, and $\mathcal{F}'$ be the homogeneous *linked* abecedarian formula computing $\text{linked\_CHSYM}_{n,d'}(\mathbf{x})$ of size s'. We think of the variable $x_{a,b}$ in $\text{linked\_CHSYM}_{n,d'}(\mathbf{x})$ as a placeholder for the sub-polynomial $\text{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$[8] of $\text{CHSYM}_{n,d}(\mathbf{x})$. Note that there is a bijection between monomials in $\text{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ and those in the polynomial we get by substituting $x_{a,b}$ in $\text{linked\_CHSYM}_{n,d'}(\mathbf{x})$ with $\text{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$.

By Observation 23, there is homogeneous abecedarian formula $\mathcal{F}_{a,b}$, of size $O(s)$ computing $\text{CHSYM}_{n,d}[a, b+1)(\mathbf{x})$ for every $a, b \in [n+1]$. Thus, if we replace every leaf of $\mathcal{F}'$ labelled by $x_{a,b}$ with $\mathcal{F}_{a,b}$, then the resulting formula is a homogeneous abecedarian formula computing $\text{CHSYM}_{n,(d\cdot d')}(\mathbf{x})$ of size $(s \cdot s')$. ◀

Finally, we observe that if we are given a homogeneous abecedarian formula computing the polynomial $\text{CHSYM}_{(n-d+1),d}(\mathbf{x})$, then we get a homogeneous multilinear formula computing the non-commutative version of $\text{ESYM}_{n,d}(\mathbf{x})$.

▶ **Observation 25.** *Consider the* elementary symmetric polynomial

$$\text{ESYM}_{n,d}(\mathbf{x}) = \sum_{1 \le i_1 < \ldots < i_d \le n} x_{i_1} \cdots x_{i_d}.$$

If there is a homogeneous abecedarian *formula computing $\text{CHSYM}_{(n-d+1),d}(\mathbf{x})$ of size s, then there is a homogeneous multilinear formula computing $\text{ESYM}_{n,d}(\mathbf{x})$ of size s.*

[8] Sum of monomials in $\text{CHSYM}_{n,d}(\mathbf{x})$ whose first variable is a and last variable is one of $\{x_a, \ldots, x_b\}$.

Proof. Suppose $\mathcal{F}$ is a homogeneous abecedarian formula computing $\mathrm{CHSYM}_{(n-d+1),d}(\mathbf{x})$ of size s. Since $\mathcal{F}$ is homogeneous, every leaf labelled by a variable can be associated with a position index. If a leaf labelled x_i has position k associated with it, then replace the label of that leaf with x_{i+k-1}. Call this formula $\mathcal{F}'$. Then clearly $\mathcal{F}'$ is a homogeneous formula of size s computing $\mathrm{ESYM}_{n,d}(\mathbf{x})$. Further note that since $\mathcal{F}$ was abecedarian, $\mathcal{F}'$ is multilinear. ◀

5.2 Proof of the Separation

We now prove Theorem 1. Let us first recall the statement.

▶ **Theorem 1** (Separating Abecedarian Formulas and Abecedarian ABPs). *Define*

$$\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x}) = \sum_{i_0=1}^{n} \left(\sum_{i_0 \le i_1 \le \ldots \le i_d \le n} x_{i_0,i_1} \cdot x_{i_1,i_2} \cdots x_{i_{d-1},i_d} \right)$$

to be the linked *complete homogeneous polynomial over n-variables of degree d. This polynomial is* abecedarian *with respect to the partition $\{X_i \ : \ i \in [n]\}$ if $X_i = \{x_{i,j} \ : \ i \le j \le n\}$.*

With respect to this partition,

1. $\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x})$ *has an* abecedarian *ABP of size $O(nd)$;*
2. *any* abecedarian *formula computing* $\mathsf{linked\_CHSYM}_{n/2,\log n}(\mathbf{x})$ *has size* $n^{\Omega(\log\log n)}$.

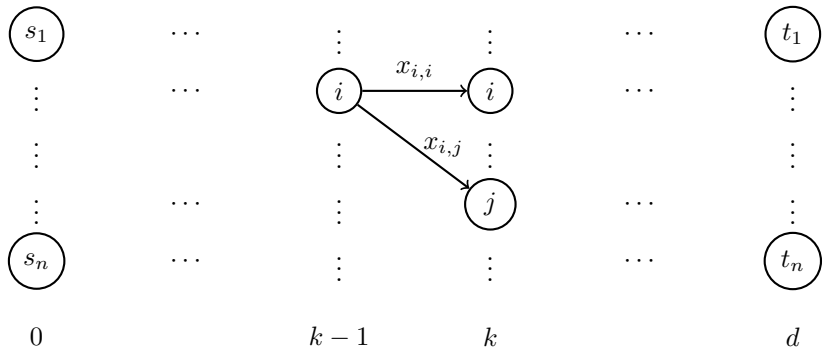
That is, there is a super-polynomial separation between abecedarian *formulas and ABPs.*

That $\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x})$ has a *small* abecedarian ABP is not very hard to see. For the lower bound, we assume that we have been given an abecedarian formula $\mathcal{F}$, computing the polynomial $\mathsf{linked\_CHSYM}_{n,\log n}(\mathbf{x})$, of size $\mathsf{poly}(n)$. We then keep making changes to this formula till we get a homogeneous multilinear formula computing $\mathrm{ESYM}_{n,n/2}(\mathbf{x})$ of size $\mathsf{poly}(n)$. Finally, we use the following theorem of Hrubeš and Yehudayoff [12] to get a contradiction.

▶ **Theorem 26** (Theorem 1, [12]). *Any homogeneous multilinear formula that computes* $\mathrm{ESYM}_{n,d}(\mathbf{x})$*, for $d \le n/2$, must have size $n \times d^{\Omega(\log d)}$.*

Let us now complete the proof of our main theorem.

Proof of Theorem 1. An abecedarian ABP of size $O(nd)$ computing $\mathsf{linked\_CHSYM}_{n,d}(\mathbf{x})$ is the following.



The ABP has $d+1$ layers, labelled 0 through d, each with n nodes. Between any consecutive layers $k-1$ and k, where $1 \le k \le d$, there is an edge from the i-th node in layer $k-1$ to the j-th node in layer k layer if $i \le j$. The label on this edge is $x_{i,j}$. All the nodes in the first layer are start nodes, and all the ones in the last layer are terminal nodes.

It is easy to check, by induction, that the polynomial computed between s_a and the b-th vertex in layer k computes $\text{CHSYM}_{n,k}[a, b+1)(\mathbf{x})$. Thus the polynomial computed by the abecedarian ABP constructed above is indeed $\text{CHSYM}_{n,d}(\mathbf{x})$, and its size is clearly $O(nd)$.

Let us now move on to proving the lower bound against abecedarian formulas. We show that there is a fixed constant ϵ_0 such that any abecedarian formula that computes the polynomial linked_$\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ must have size atleast $\Omega(n^{\epsilon_0 \log\log n})$. Suppose this is not the case. Then for every $\epsilon > 0$, there is an abecedarian formula $\mathcal{F}'(\epsilon)$ of size $O(n^{\epsilon \log\log n})$ that computes linked_$\text{CHSYM}_{n/2,\log n}(\mathbf{x})$.

Without loss of generality, we can assume that $\mathcal{F}'(\epsilon)$ has fan-in 2. Further, by Lemma 6, we can reduce the depth of $\mathcal{F}'(\epsilon)$ to log-depth. That is, we get an abecedarian formula $\mathcal{F}'_1(\epsilon)$ computing linked_$\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ of depth $O(\epsilon \log n \log\log n)$ and size $O(n^{c_1 \epsilon \log\log n})$. Here c_1 is a fixed constant independent of ϵ.

Next, since the degree of the polynomial being computed is *small*, Lemma 5 implies that $\mathcal{F}'_1(\epsilon)$ can in fact be homogenised without much blow-up in size. In other words, there is a homogeneous abecedarian formula computing linked_$\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ of size $O(n^{c_1 c_2 \epsilon \log\log n})$, where c_2 is again a fixed constant independent of ϵ. Let this formula be $\mathcal{F}'_2(\epsilon)$.

By Observation 21, we can then use $\mathcal{F}'_2(\epsilon)$ to get a homogeneous linked abecedarian formula $\mathcal{F}'_3(\epsilon)$ of size $O(n^{c_1 c_2 \epsilon \log\log n})$ that computes the same polynomial. Further, because of Observation 22, we know that there is a homogeneous abecedarian formula, say $\mathcal{F}$, of size $\text{poly}(n) = O(n^{c_1 c_2 \epsilon \log\log n})$ that computes $\text{CHSYM}_{n/2,\log n}(\mathbf{x})$.

With $\mathcal{F}$ and $\mathcal{F}'_3(\epsilon)$ in hand, we get a homogeneous abecedarian formula $\text{CHSYM}_{n/2,\log^2 n}(\mathbf{x})$ because of Lemma 24. To get such a formula for $\text{CHSYM}_{n/2,n/2}(\mathbf{x})$, we need to use Lemma 24 at most k times where

$$(\log n)^k = \frac{n}{2} \implies k = O\left(\frac{\log n}{\log\log n}\right).$$

Thus, using Lemma 24 repeatedly at most $O(\log n / \log\log n)$ times, we get that there is a homogeneous abecedarian formula, $\mathcal{F}(\epsilon)$, computing $\text{CHSYM}_{n/2,n/2}(\mathbf{x})$ of size

$$O(n^{(c_1 c_2 \epsilon \log\log n)\cdot(\log n/\log\log n)}) = O(n^{(c_1 c_2 \epsilon \log n)}).$$

By Observation 25, we know that $\mathcal{F}(\epsilon)$ can be used to get a homogeneous multilinear formula, $\mathcal{F}_1(\epsilon)$, computing $\text{ESYM}_{n-1,n/2}(\mathbf{x})$ of size $O(n^{(c_1 c_2 \epsilon \log n)})$. Finally, Theorem 26 tells us that there is a constant δ such that any homogeneous multilinear formula computing $\text{ESYM}_{n-1,n/2}(\mathbf{x})$ must have size at least $n^{\delta \cdot \log n}$. For $\epsilon = \delta/2c_1c_2$, this contradicts the existence of $\mathcal{F}_1(\epsilon)$ and hence $\mathcal{F}'(\epsilon)$. Thus, it must be the case that any abecedarian formula computing linked_$\text{CHSYM}_{n/2,\log n}(\mathbf{x})$ has size at least $n^{\Omega(\log\log n)}$. This completes the proof. ◀

6 Proofs of the Remaining Statements

In this section we give proof ideas of the remaining statements mentioned in the introduction.

6.1 Formula Lower Bounds from Structured Formula Lower Bounds

▶ **Corollary 3.** *Let the polynomial* linked_$\text{CHSYM}_{n,d}(\mathbf{x})$ *be as defined in Theorem 1. An* $n^{\omega(1)}$ *lower bound against* abecedarian *formulas for* linked_$\text{CHSYM}_{\log n,n}(\mathbf{x})$ *would imply a super-polynomial separation between non-commutative ABPs and formulas.*

Proof. By Theorem 1, we know that the ABP complexity of linked_CHSYM$_{\log n,n}(\mathbf{x})$ is poly(n). Therefore any formula computing the polynomial must have size at least $n^{\Omega(1)}$. Further, note that the polynomial is abecedarian with respect to a partition of size $O(\log n)$. Therefore, by Theorem 2, if there is a formula $\mathcal{F}$ computing linked_CHSYM$_{\log n,n}(\mathbf{x})$ of size s, then there is an abecedarian formula computing it of size poly(s). This immediately implies the given statement. ◀

▶ **Corollary 4.** *Let* linked_CHSYM$_{n,d}(\mathbf{x})$ *be as defined in Theorem 1. An* $n^{\omega(1)}$ *lower bound against homogeneous formulas for* linked_CHSYM$_{n,\log n}(\mathbf{x})$ *would result in a super-polynomial separation between ABPs and formulas in the non-commutative setting.*

Proof. By Theorem 1, we know that the ABP complexity of linked_CHSYM$_{n,\log n}$ is poly(n). Further, the degree of the polynomial is $O(\log n)$. Thus, by Lemma 5, if there is a formula computing linked_CHSYM$_{n,\log n}(\mathbf{x})$ of size s, then there is a homogeneous formula computing it of size poly(s). This immediately implies the given statement. ◀

▶ **Corollary 11.** *An* $n^{\omega(1)}$ *lower bound against homogeneous formulas computing the* n*-variate iterated matrix multiplication polynomial of degree* $\log n$*,* IMM$_{n,\log n}(\mathbf{x})$*, implies a super-polynomial separation between ABPs and formulas in the non-commutative setting.*

Proof. Clearly, the ABP complexity of IMM$_{n,\log n}(\mathbf{x})$ is poly(n). Thus, by Lemma 5, if there is a formula computing IMM$_{n,\log n}(\mathbf{x})$ of size s, then there is a homogeneous formula computing it of size poly(s). This immediately implies the given statement. ◀

6.2 Known Relations in the Non-Commutative Setting that Continue to Hold with the Abecedarian Restriction

▶ **Observation 9** (The Usual Inclusions)**.** *Let* abc - VP$_{nc}$*,* abc - VBP$_{nc}$ *and* abc - VF$_{nc}$ *denote the classes of* abecedarian *polynomials over* n *variables that can be computed by* poly(n) *sized* abecedarian *circuits,* abecedarian *ABPs and* abecedarian *formulas respectively. Then,*

$$\text{abc - VF}_{nc} \subseteq \text{abc - VBP}_{nc} \subseteq \text{abc - VP}_{nc}.$$

Proof. Suppose $f \in$ abc - VF$_{\text{nc}}$. Then f is abecedarian, and in particular $f \in$ VF$_{\text{nc}}$. But we know that VF$_{\text{nc}} \subseteq$ VBP$_{\text{nc}}$, and so $f \in$ VBP$_{\text{nc}}$. By Observation 8, this implies that $f \in$ abc - VBP$_{\text{nc}}$.

Similarly, suppose $f \in$ abc - VBP$_{\text{nc}}$. Then f is abecedarian, and $f \in$ VBP$_{\text{nc}}$. But VBP$_{\text{nc}} \subseteq$ VP$_{\text{nc}}$, and so $f \in$ VP$_{\text{nc}}$. By Observation 7, this implies that $f \in$ abc - VP$_{\text{nc}}$. ◀

▶ **Observation 10** (Converting Abecedarian ABPs into Abecedarian Formulas)**.** *Suppose* f *is an* abecedarian *polynomial of degree* d*. If there is an* abecedarian *ABP* $\mathcal{A}$ *of size* s *computing it, then there is an* abecedarian *formula* $\mathcal{F}$ *computing* f *of size* $O(s^{\log d})$*.*

Proof. The formula we get using the usual divide-and-conquer algorithm has the property that polynomials computed at any of its gate is a polynomial computed between two vertices in the ABP. Thus by definition of abecedarian ABPs, the statement follows via the usual algorithm. ◀

References

1 V. Arvind, P. S. Joglekar, and S. Raja. Noncommutative valiant's classes: Structure and complete problems. *ACM Trans. Comput. Theory*, 9(1), 2016. doi:10.1145/2956230.

2 Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. *Comput. Complex.*, 27(1):1–29, 2018. doi:10.1007/s00037-016-0148-5.

3 Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992. doi:10.1137/0221006.

4 Richard P. Brent. The parallel evaluation of general arithmetic expressions. *Journal of the ACM*, 21(2):201–206, 1974. doi:10.1145/321812.321815.

5 Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness amplification for non-commutative arithmetic circuits. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC*, volume 102 of *LIPIcs*, pages 12:1–12:16, 2018. doi:10.4230/LIPIcs.CCC.2018.12.

6 Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. A quadratic lower bound for algebraic branching programs and formulas. *CoRR*, 1911.11793v2, 2019. arXiv:1911.11793v2.

7 Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating multilinear branching programs and formulas. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 615–624. ACM, 2012. doi:10.1145/2213977.2214034.

8 Nathanaël Fijalkow, Guillaume Lagarde, Pierre Ohlmann, and Olivier Serre. Lower bounds for arithmetic circuits via the hankel matrix. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 154 of *LIPIcs*, pages 24:1–24:17, 2020. doi:10.4230/LIPIcs.STACS.2020.24.

9 Pavel Hrubes and Avi Wigderson. Non-commutative arithmetic circuits with division. *Theory Comput.*, 11:357–393, 2015. doi:10.4086/toc.2015.v011a014.

10 Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Relationless completeness and separations. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC*, pages 280–290. IEEE Computer Society, 2010. doi:10.1109/CCC.2010.34.

11 Pavel Hrubeš, Avi Wigderson, and Amir Yehudayoff. Non-commutative circuits and the sum-of-squares problem. *Journal of the American Mathematical Society*, 24(3):871–898, 2011. URL: https://www.ams.org/journals/jams/2011-24-03/S0894-0347-2011-00694-2/S0894-0347-2011-00694-2.pdf.

12 Pavel Hrubes and Amir Yehudayoff. Homogeneous formulas and symmetric polynomials. *Comput. Complex.*, 20(3):559–578, 2011. doi:10.1007/s00037-011-0007-3.

13 L. Hyafil. The power of commutativity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 171–174, 1977. doi:10.1109/SFCS.1977.31.

14 K. Kalorkoti. A lower bound for the formula size of rational functions. *SIAM J. Comput.*, 14(3):678–687, 1985. doi:10.1137/0214050.

15 Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower bounds and PIT for non-commutative arithmetic circuits with restricted parse trees. *Comput. Complex.*, 28(3):471–542, 2019. doi:10.1007/s00037-018-0171-9.

16 Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. Non-commutative computations: lower bounds and polynomial identity testing. *Chic. J. Theor. Comput. Sci.*, 2019, 2019. URL: http://cjtcs.cs.uchicago.edu/articles/2019/2/contents.html.

17 Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Theory of Computing*, 12(1):1–38, 2016. doi:10.4086/toc.2016.v012a012.

18 Merriam and Webster. Definition of abecedarian. Word of the Day at www.merriam-webster.com, 2019. URL: https://www.merriam-webster.com/word-of-the-day/abecedarian-2019-03-06.

19 Eduard Ivanovich Nechiporuk. On a boolean function. *Dokl. Akad. Nauk SSSR*, 169:765–766, 1966. URL: http://mi.mathnet.ru/dan32449.

20 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 410–418. ACM, 1991. doi:10.1145/103418.103462.

21 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40:1–40:15, 2013. doi:10.1145/2535928.

22 Ramprasad Saptharishi and Anamay Tengse. Quasipolynomial hitting sets for circuits with restricted parse trees. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 122 of *LIPIcs*, pages 6:1–6:19, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.6.

23 Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing*, pages 249–261. ACM, 1979. doi:10.1145/800135.804419.