INSTITUTE OF MATHEMATICS

THE CZECH ACADEMY OF SCIENCES

Monotone classes beyond VNP

Prerona Chatterjee
Kshitij Gajjar
Anamay Tengse

Preprint No. 52-2022

PRAHA 2022

Monotone Classes Beyond VNP

Prerona Chatterjee [*] Kshitij Gajjar [†] Anamay Tengse [‡]

September 26, 2022

Abstract

We study the natural monotone analogues of various equivalent definitions of VPSPACE: a well studied class [Poi08, KP09, Mal11, MR13] that is believed to be larger than VNP. We show an exponential separation between the monotone version of Poizat's definition of VPSPACE [Poi08] and monotone VNP. We also show that unlike their non-monotone counterparts, these monotone analogues are not equivalent, with exponential separations in some cases.

The primary motivation behind our work is to understand the monotone complexity of *transparent polynomials*, a concept that was recently introduced by Hrubeš and Yehudayoff [HY21]. In that context, we are able to show that *transparent polynomials* of large sparsity are hard for the monotone analogues of all definitions of VPSPACE, except for the one due to Poizat.

[*]Institute of Mathematics of the Czech Academy of Sciences. Research supported by Grant GX19-27871X of the Czech Science Foundation. Part of this work was done as a PhD student at TIFR, Mumbai (partially supported by a Google PhD fellowship). Email: prerona.ch@gmail.com.

[†]Indian Institute of Technology Jodhpur, Rajasthan, India – 342037. Part of this work was done as a postdoc at NUS, Singapore (NUS ODPRT Grant WBS No. R-252-000-A94-133). Email: kshitij@iitj.ac.in.

[‡]Department of Computer Science, University of Haifa, Israel. Research supported by the Israel Science Foundation (grant No. 716/20). Email: anamay.tengse@gmail.com.

1 Introduction

The aim of algebraic complexity is to classify polynomials in terms of how hard it is to compute them, and the most standard model for computing polynomials is that of an *algebraic circuit*. An algebraic circuit is a rooted, directed acyclic graph where the leaves are labelled with variables or field constants and internal nodes are labelled with addition $(+)$ or multiplication $(\times)$. Every node therefore naturally computes a polynomial and the polynomial computed by the root is said to be the polynomial computed by the circuit. Definition 2.1 is a formal definition.

The central question in the area is to show super-polynomial lower bounds against algebraic circuits for *explicit* polynomials, or to show that $\mathsf{VP} \neq \mathsf{VNP}$, the algebraic analogue of the famed P vs. NP question. However, proving strong lower bounds against circuits has turned out to be a difficult problem. Much of the research therefore naturally focusses on various restricted algebraic models which compute correspondingly structured polynomials.

One such syntactic restriction is that of *monotonicity*, where the models are not allowed to use any negative constants. Therefore, trivially, monotone circuits always compute polynomials with only non-negative coefficients. Such polynomials are called *monotone polynomials*. We denote the class of all polynomials that are efficiently computable by monotone algebraic circuits by mVP. Also note that any monomial computed during intermediate computation in a monotone circuit can never get cancelled out, making it a fairly weak model. As a result, several strong lower bounds are known against monotone circuits.

Lower bounds in the monotone setting There has been a long line of classical works that prove lower bounds against monotone algebraic circuits [Sch76, SS77, SS80, JS82, Kuz85, KZ86, Gas87]. The most well-known among these, is the result of Jerrum and Snir [JS82] where they showed exponential lower bounds against monotone circuits for many polynomial families, including the Permanent (Perm_n). In particular, they showed that every monotone algebraic circuit computing the n^2-variate Perm_n must have size at least $2^{\Omega(n)}$. A few of the more recent works on monotone lower bounds include [RY11, GS12, CKR20].

Additionally, many separations that are believed to be true in the general setting have actually been proved to be true in the monotone setting [SS77, HY16, Yeh19, Sri19]. Most remarkably, Yehudayoff [Yeh19] showed an exponential separation between the computational powers of the the monotone analogues of VP and VNP (denoted by mVP and mVNP respectively).

Another line of work in this setting tries to understand the power of non-monotone computational models while computing monotone polynomials. Valiant [Val80], in his seminal paper, showed that there is a family of monotone polynomials which can be computed by polynomial sized non-monotone algebraic circuits such that any monotone algebraic circuit computing them must have exponential size. More recent works [HY13, CDM21, CDGM22, CGM22] have shown even stronger separations between the relative powers of monotone and non-monotone models

while computing monotone polynomials.

Newton polytopes, transparency and monotone complexity Returning briefly to the general setting, an interesting conjecture relating the algebraic complexity of a bivariate polynomial to its geometric property, is the 'Tau-conjecture' (also written as τ conjecture). The Newton polytope of an n-variate polynomial f, denoted by $\mathsf{Newt}(f)$, is the convex hull in $\mathbb{R}^n$ of the *exponent vectors* of the monomials in the support of f. Recently, Hrubeš and Yehudayoff [HY21] proposed the notion of *Shadows of Newton polytopes* (the maximum number of vertices in any linear projection of the polytope to a plane) as an approach to refute the τ-conjecture for Newton polygons made by Koiran, Portier, Tavenas and Thomassé [KPTT15].

Informally, the τ-conjecture for Newton polygons [KPTT15] states that if f is a bivariate polynomial that can be written as an s-sum of r-products of p-sparse polynomials, then its Newton polygon has at most $\mathrm{poly}(s, r, p)$ vertices. Definition 2.4 and Conjecture 2.5 are the formal definition of Newton polytopes and the formal statement of the τ-conjecture for Newton polygons respectively.

This is a fairly strong conjecture and it implies, among other things, that $\mathsf{VP} \neq \mathsf{VNP}$. However, observe that the Newton polygon retains no information about the coefficients of the polynomial. Since the algebraic complexity of polynomials is believed to be heavily dependent on coefficients (for example the determinant (Det_n) is efficiently computable by algebraic circuits and this is expected to not be the case for Perm_n even though they have the same set of monomials), the τ-conjecture for Newton polygons is believed to be false.

The approach suggested by Hrubeš and Yehudayoff [HY21] used shadows of Newton polytopes as a means to move from the multivariate setting to the bivariate setting, and use polynomials like determinant (Det_n) to refute the conjecture. The difficulty in this strategy however, is to find a polynomial in VP that exhibits high *shadow complexity*, since even when a candidate polynomial is fixed, say Det_n, it is not easy to design a suitable bivariate projection.

As a means to tackle this issue, Hrubeš and Yehudayoff introduced the notion of *transparent polynomials* — polynomials that can be projected to bivariates in such a way that all of their monomials become vertices of the resulting Newton polygon. Further, they also gave examples of polynomials with exponentially large sets of monomials that are provably transparent. Therefore a proof of any one of these polynomials being in VP would directly refute the τ-conjecture for Newton polytopes.

Even though Hrubeš and Yehudayoff [HY21] were not able to actually use this approach to refute the conjecture, they used the notions of shadows and transparency to come up with yet another method for proving lower bounds against monotone algebraic circuits. They showed that the monotone circuit complexity of a polynomial is lower bounded by its shadow complexity when the polynomial is transparent.

Theorem 1.1 ([HY21, Theorem 2]). *If f is transparent then every monotone circuit computing f has size at least $\Omega(|\mathrm{supp}(f)|)$.*

As a corollary, they present an n-variate polynomial such that any monotone algebraic circuit computing it must have size $\Omega(2^{n/3})$.

1.1 Our Contribution

Here we state our contributions informally; the formal statements can be found in Section 3. The goal of this work is two-fold.

The first goal is to understand how restrictive the notion of transparency is. Our search begins with an observation by Yehudayoff [Yeh19], that any lower bound against mVP depending solely on the support of the hard polynomial, automatically "lifts" to mVNP with the same parameters[1]. Since transparency is a property solely of the Newton polytope, and hence of the support of the polynomial, the above observation shows that any transparent polynomial that is non-sparse (has super-polynomially large support) is hard to compute even for mVNP. However we believe that transparency is a very strong property and a natural question for us therefore, is whether there are even larger classes of monotone polynomials that do not contain non-sparse, transparent polynomials.

This brings us to the second goal of this work — studying monotone models of computation that can possibly compute polynomials outside even mVNP. Classes larger than VNP had not been defined in the monotone world prior to this work, and we therefore turn to the literature in the non-monotone setting. Here, VPSPACE is a well studied class [Poi08, KP09, Mal11, MR13] that is believed to be strictly larger than VNP. Interestingly there are multiple definitions of VPSPACE, resulting from varied motivations, which are all known to be essentially equivalent [Mal11, MR13].

We study the natural monotone analogues of these definitions and show that unlike the non-monotone setting, the powers of the different resulting models varies greatly. This allows us to then analyse the technique of Hrubeš and Yehudayoff against monotone classes that are possibly larger than mVNP. The following figure succinctly describes our main results.

In Figure 1, the node labels refer to the following classes of polynomial families that have $\mathrm{poly}(n)$ complexity under various models, as follows:

- msuccABP - monotone succinct ABPs (Definition 3.1),
- $\mathsf{mVP}_{\mathtt{quant}}$ - quantified monotone circuits (Definition 3.3),
- $\mathsf{mVP}_{\mathtt{sum,prod}}$ - monotone circuits with summation and production gates (Definition 3.7),
- $\mathsf{mVP}_{\mathtt{proj}}$ - monotone circuits with projection gates (Definition 3.9).

[1] [Yeh19]: "If a monotone circuit-size lower bound for $q(\mathbf{x})$ holds also for all polynomials that are equivalent to $q(\mathbf{x})$ then it also holds for every mVNP circuit computing $q(\mathbf{x})$."

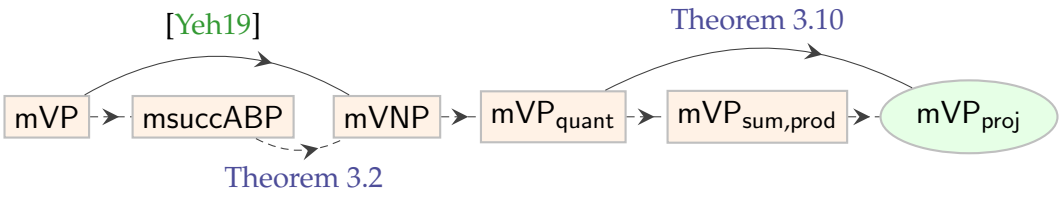

Figure 1: Nodes represent classes of polynomial families; $A \dashrightarrow B$ denotes $A \subseteq B$ and $A \longrightarrow B$ denotes $A \subsetneq B$.

The orange, rectangular nodes denote the classes in which sparsity of transparent polynomials in it is bounded by a constant factor of the size of the smallest $\mathcal{M}$ computing it, if $\mathcal{M}$ is the computational model corresponding to the class (Theorem 3.8). An interesting point to note here is that there is an exponential separation between $\mathsf{mVP}_{\mathsf{quant}}$ and $\mathsf{mVP}_{\mathsf{proj}}$, which means that at least one of the inclusions: $\mathsf{mVP}_{\mathsf{quant}}$ to $\mathsf{mVP}_{\mathsf{sum,prod}}$, and $\mathsf{mVP}_{\mathsf{sum,prod}}$ to $\mathsf{mVP}_{\mathsf{proj}}$ is strict with an exponential separation.

1.2 Organisation of the paper

We begin in Section 2 with formal definitions for all the models of computation that we will be using. Next, we define the monotone analogues of the various definitions of VPSPACE, and outline our results about them in Section 3. The proofs of our results are discussed in Section 4, Section 5, Section 6 and Section 7. We conclude with Section 8, where we discuss some of the important open threads from our work.

2 Preliminaries

We shall use the following notation for the rest of the paper.

- We use the standard shorthand $[n] = \{1, 2, \ldots, n\}$.
- We use boldface letters like $\mathbf{x}, \mathbf{z}, \mathbf{e}$ to denote tuples/sets of variables or constants, individual members are expressed using indexed version of the usual symbols: $\mathbf{e} = (e_1, e_2, \ldots, e_n)$, $\mathbf{x} = \{x_1, \ldots, x_n\}$. We also use $|\mathbf{y}|$ to denote the size/length of a vector $\mathbf{y}$.

  For vectors $\mathbf{x}$ and $\mathbf{e}$ of the same length n, we use the shorthand $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$.
- For a polynomial $f(\mathbf{x})$ and a monomial $m = \mathbf{x}^{\mathbf{e}}$, we refer to the coefficient of m in f by $\mathrm{coeff}_f(m)$. The support $\mathrm{supp}(f)$ of a polynomial f is given by $\{m : \mathrm{coeff}_f(m) \neq 0\}$, and the *sparsity* of a polynomial is the size of its support, $|\mathrm{supp}(f)|$.

Algebraic Circuits and basic monotone classes We now formally define algebraic circuits.

Definition 2.1 (Algebraic circuits)**.** *An algebraic circuit is a directed acyclic graph with leaves (nodes with in-degree zero) labelled by formal variables and constants from the field, and other nodes labelled by addition* $(+)$ *and multiplication* $(\times)$*.*

The leaves compute their labels, and every other node computes the operation it is labelled by, on the polynomials along its incoming edges. There is a unique node of out-degree zero called the root, and the circuit is said to compute the polynomial computed at the root.

The size *of a circuit,* $\mathcal{C}$*, denoted by* $\text{size}(\mathcal{C})$*, is the number of nodes in the graph.*

An algebraic circuit over $\mathbb{Q}$ *or* $\mathbb{R}$ *is said to be* monotone*, if all the constants appearing in it are non-negative.* ◊

Next, we formally define the relevant classes of monotone polynomials that are already present in literature, namely the monotone analogues of VP and VNP.

Definition 2.2 (Monotone VP (mVP))**.** *A family* $\{f_n\}$ *of monotone polynomials is said to be in* mVP*, if there exists a constant* $c \in \mathbb{N}$ *such that for all large* n*,* f_n *depends on at most* n^c *variables, has degree at most* n^c*, and is computable by a monotone algebraic circuit of size at most* n^c*.* ◊

Definition 2.3 (Monotone VNP (mVNP))**.** *A family* $\{f_n\}$ *of monotone polynomials is said to be in* mVNP*, if there exists a constant* $c \in \mathbb{N}$*, and a family* $\{g_m\} \in \mathsf{mVP}$*, such that for all large enough* n*, and* $m \leq n^c$*,* f_n *satisfies the following.*

$$f_n(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^{|\mathbf{y}|}} g_m(\mathbf{x}, \mathbf{y} = \mathbf{a})$$ ◊

Newton Polytopes and the tau conjecture for Newton polygons

Definition 2.4 (Newton polytopes)**.** *For a polynomial* $f(\mathbf{x})$*, its Newton polytope* $\mathsf{Newt}(f) \subseteq \mathbb{R}^n$*, is defined as the convex hull of the* exponent vectors *of the monomials in its support.*

$$\mathsf{Newt}(f) := \text{conv}\left(\{\mathbf{e} : \mathbf{x}^{\mathbf{e}} \in \text{supp}(f)\}\right)$$

A point $\mathbf{e} \in \mathsf{Newt}(f)$ *is said to be a* vertex*, if it cannot be written as a convex combination of* other *points in* $\mathsf{Newt}(f)$*. We denote the set of all vertices of a polytope* $\mathcal{P}$ *using* $\text{vert}(\mathcal{P})$*.* ◊

Conjecture 2.5 (τ conjecture for Newton polytopes [KPTT15])**.** *Suppose* $f(x, y)$ *is a bivariate polynomial that can be written as* $\sum_{i \in [s]} \prod_{j \in [r]} T_{i,j}(x, y)$*, where each* $T_{i,j}$ *has sparsity at most* p*. Then the Newton polygon of* f *has* $\text{poly}(s, r, p)$ *vertices.*

We now move on to formally stating the various definitions of VPSPACE. This will allow us to then define their monotone analogues.

2.1 Various definitions of VPSPACE

Koiran and Perifel [KP09, KP07] were the first to define VPSPACE as the class of polynomials (of degree that is potentially exponential in the number of underlying variables) whose coefficients can be computed in PSPACE/ poly and $\mathsf{VPSPACE}_b$ to be the polynomials in VPSPACE that have degree bounded by a polynomial in the number of underlying variables. They showed that if $\mathsf{VP} \neq \mathsf{VPSPACE}_b$ then either $\mathsf{VP} \neq \mathsf{VNP}$ or $\mathsf{P/poly} \neq \mathsf{PSPACE/poly}$.

Later, Poizat [Poi08] gave an alternate definition that does not rely on any boolean machinery, but instead uses a new type of gate called a *projection gate*.

Definition 2.6 (Projection gates [Poi08])**.** *A* projection *gate is a* unary *gate that is labelled by a variable z and a constant $b \in \{0,1\}$, denoted by $\mathtt{fix}_{(z=b)}$. It returns the partial evaluation of its input polynomial, at $z = b$, that is, $\mathtt{fix}_{(z=b)}(f(z,\mathbf{x})) = f(b,\mathbf{x})$.* ◊

Poizat defined algebraic circuits with projection gates and then defined VPSPACE to be the class of polynomial families that are efficiently computable by this model. Poizat showed[2] that this definition is equivalent to that of Koiran and Perifel.

Definition 2.7 (Algebraic circuits with projection gates [Poi08])**.** *An* algebraic circuit with projection gates *is a directed acyclic graph with leaves (nodes with in-degree zero) labelled by formal variables and constants from the field, and other nodes labelled by addition $(+)$, multiplication $(\times)$ or projection $(\mathtt{fix}_{(z=b)})$. The leaves compute their labels, the nodes labelled by addition and multiplication compute the operation they are labelled by, on the polynomials along its incoming edges, and nodes labelled by projection gates compute the polynomial described in Definition 2.6. There is a unique node of out-degree zero called the root, and the circuit is said to compute the polynomial computed at the root.*

The size *of an algebraic circuit with projection gates is the number of nodes in the graph.* ◊

Adding to Poizat's work, Malod [Mal11] characterised VPSPACE using exponentially large *algebraic branching programs (ABPs)* that are *succinct*. Malod's work defines the *complexity* of an ABP as the size of the smallest algebraic circuit that encodes its graph — outputs the corresponding edge label when given the two endpoints as input. An n-variate ABP is then said to be *succinct*, if its complexity is $\mathrm{poly}(n)$.

Definition 2.8 (Succinct ABPs [Mal11])**.** *A* succinct ABP *over the n variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a three tuple $(B, \mathbf{s}, \mathbf{t})$ with $|\mathbf{s}| = |\mathbf{t}| = r$, where*

- *$\mathbf{s}$ is the label of the source vertex, and $\mathbf{t}$ is the label of the sink(target) vertex.*
- *$B(\mathbf{u}, \mathbf{v}, \mathbf{x})$ is an algebraic circuit that describes a directed acyclic graph G_B on the vertex set $\{0,1\}^r$ in the following way. For any two vertices $\mathbf{a}, \mathbf{b} \in \{0,1\}^r$, the output $B(\mathbf{u} = \mathbf{a}, \mathbf{v} = \mathbf{b}, \mathbf{x})$ is the label of the edge from $\mathbf{a}$ to $\mathbf{b}$ in the ABP.*

[2] The work of Poizat is written in French, Malod [Mal11] provides an alternate exposition of some of the main results in English.

The polynomial computed by the ABP is the sum of polynomials computed along all $\mathbf{s}$ *to* $\mathbf{t}$ *paths in* G_B*; where each path computes the product of the labels of the constituent edges.*

The size of the circuit B *is said to be the* complexity *of the succinct ABP. The number of vertices* 2^r *is the* size *of the succinct ABP, and the length of the longest* $\mathbf{s}$ *to* $\mathbf{t}$ *path is called the* length *of the ABP.* $\diamondsuit$

In the same work [Mal11], Malod alternatively characterised VPSPACE using an interesting algebraic model that resembles *(totally) quantified boolean formulas* that are known to characterise PSPACE. This model, which we refer to as "quantified algebraic circuits", is defined using special types of projection gates called *summation* and *production* gates.

Definition 2.9 (Summation and Production gates [Mal11])**.** Summation *and* production *gates are unary gates that are labelled by a variable* z*, and are denoted by* $\mathtt{sum}_z$ *and* $\mathtt{prod}_z$ *respectively. A summation gate returns the sum of the* $(z = 0)$ *and* $(z = 1)$ *evaluations of its input, and a production gate returns the product of those evaluations. That is,* $\mathtt{sum}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) + f(1, \mathbf{x})$*, and* $\mathtt{prod}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) \cdot f(1, \mathbf{x})$.

We sometimes use $\mathtt{sum}_{\{z_1,\ldots,z_k\}}$ *to refer to the nested expression* $\mathtt{sum}_{z_1} \cdots \mathtt{sum}_{z_k}$ *(similarly for* $\mathtt{prod}$*); it can be checked that the order does not matter here.* $\diamondsuit$

A quantified algebraic circuit has the form $\mathtt{Q}^1_{z_1}\mathtt{Q}^2_{z_2}\cdots\mathtt{Q}^m_{z_m}\mathcal{C}(\mathbf{x}, \mathbf{z})$, where each $\mathtt{Q}^i$ is a summation or a production, and $\mathcal{C}(\mathbf{x}, \mathbf{z})$ is a usual algebraic circuit.

Definition 2.10 (Quantified Algebraic Circuits [Mal11])**.** *A quantified algebraic circuit is an algebraic circuit has the form*

$$\mathtt{Q}^{(1)}_{z_1}\mathtt{Q}^{(2)}_{z_2}\cdots\mathtt{Q}^{(m)}_{z_m}\mathcal{C}(\mathbf{x}, \mathbf{z})$$

where $|\mathbf{z}| = m$*,* $\mathtt{Q}^{(i)} \in \{\mathtt{sum}, \mathtt{prod}\}$ *for each* $i \in [m]$*, and* $\mathcal{C}$ *is an algebraic circuit. The size of a quantified algebraic circuit is* $m + \text{size}(\mathcal{C})$*.* $\diamondsuit$

Finally, Mahajan and Rao [MR13] defined algebraic analogues of small space computation (e.g. L, NL) using the notion of *width* of an algebraic circuit. They use their definitions to import some of the relationships from the boolean world to the algebraic world (e.g, they show VL $\subseteq$ VP). They further show that their definition of uniform polynomially-bounded-space computation coincides with that of uniform-VPSPACE as defined by Koiran and Perifel [KP09].

We now narrow our focus to the definitions due to Poizat [Poi08] and Malod [Mal11]. We choose these definitions because they are algebraic in nature, and have fairly natural monotone analogues. We elaborate a bit more about this decision in Appendix A.

Remark. *It should be noted that all the above mentioned definitions of* VPSPACE *allow for the polynomial families to have large degree — as high as* $\exp(\text{poly}(n))$*. The main focus of our work, however, is to compare the monotone analogues of these models with* mVP *and* mVNP*. Since the latter classes only contain low-degree polynomials, we will only work with polynomials of degree* $\text{poly}(n)$*, or* VPSPACE_b *as defined in [KP09], for the rest of this paper.* $\diamondsuit$

3 Monotone analogues of VPSPACE, and our contributions

We now define monotone analogues for the various definitions of VPSPACE outlined in the previous section, and compare the powers of the resulting monotone models/classes.

Monotone succinct ABPs. We first consider the natural monotone analogue of the definition due to Malod [Mal11] which uses succinct algebraic branching programs (Definition 2.8).

Malod showed that every family $\{f_n\}$ in VPSPACE can be computed by $2^{\text{poly}(n)}$ sized ABPs that have *complexity* $\text{poly}(n)$. Recall that the complexity of a succinct ABP is the size of the smallest algebraic circuit that encodes its graph.

We therefore define monotone succinct ABPs as ABPs that can be succinctly described by *monotone* algebraic circuits of size $\text{poly}(n)$. However this restriction forces that if the monomial $\mathbf{x}^{\mathbf{e}}$ appears in any edge-label $(\mathbf{a}, \mathbf{b})$, then it also appears in the label of $(\bar{1}, \bar{1})$. Therefore self-loops are inevitably present in succinct ABPs in the monotone setting. To handle this, we additionally allow the *length* of the ABP, say ℓ, to be predefined[3] so that now the polynomial computed by the ABP can be defined to be the sum of polynomials computed by all $\mathbf{s}$ – $\mathbf{t}$ paths of length at most ℓ.

Definition 3.1 (Monotone Succinct ABPs)**.** *A* monotone succinct ABP *over* $\mathbf{x} = \{x_1, \ldots, x_n\}$ *is a four tuple* $(B, \mathbf{s}, \mathbf{t}, \ell)$ *with* $|\mathbf{s}| = |\mathbf{t}| = r$*, where*

- ℓ *is the* length *of the ABP.*
- $\mathbf{s}$ *is the label of the source vertex, and* $\mathbf{t}$ *is the label of the sink(target) vertex.*
- $B(\mathbf{u}, \mathbf{v}, \mathbf{x})$ *is a* monotone *algebraic circuit that describes a directed graph* G_B *on the vertex set* $\{0,1\}^r$ *in the following way. For any two vertices* $\mathbf{a}, \mathbf{b} \in \{0,1\}^r$*, the output* $B(\mathbf{u} = \mathbf{a}, \mathbf{v} = \mathbf{b}, \mathbf{x})$ *is the label of the edge from* $\mathbf{a}$ *to* $\mathbf{b}$ *in the ABP.*

The polynomial computed by the ABP is the sum of polynomials computed along all $\mathbf{s}$ *to* $\mathbf{t}$ *paths in* G_B *of length at most* ℓ*; where each path computes the product of the labels of the constituent edges.*

The size of the circuit B *is said to be the* complexity *of the monotone succinct ABP. The number of vertices* 2^r *is the* size *of the succinct ABP.* ♢

Note that since B is a monotone algebraic circuit, all the edge-labels in the ABP are monotone polynomials over $\mathbf{x}$. It is also not hard to see that any polynomial $f \in$ mVP is computable by this model. If $\mathcal{C}$ is the monotone circuit computing f, then the monotone succinct ABP computing f is $(\mathcal{C}', 0, 1, 1)$ where $\mathcal{C}'(u, v, \mathbf{x}) = v \cdot \mathcal{C}(\mathbf{x})$.

However, surprisingly, we show that the computational power of monotone succinct ABPs when computing polynomials of *bounded degree* does not go beyond mVNP.

[3] It is not hard to see that the analogous definition in the non-monotone setting is equivalent to Malod's definition (Definition 2.8). This is essentially because of the connection to Iterated Matrix Multiplication.

Theorem 3.2. *If an n-variate polynomial $f(\mathbf{x})$ of degree $\mathrm{poly}(n)$ is computable by a monotone succinct ABP of complexity $\mathrm{poly}(n)$, then $f(\mathbf{x}) \in \mathsf{mVNP}$.*

In contrast, Malod [Mal11] showed that every family in VPSPACE admits succinct ABPs of polynomial complexity and we expect $\mathsf{VPSPACE}_b$ to be a much bigger class than VNP.

Quantified monotone circuits. As mentioned earlier, Malod [Mal11] had also characterised the class VPSPACE using the notion of quantified algebraic circuits (Definition 2.10). We now consider its natural monotone analogue, which we call quantified monotone circuits.

Definition 3.3 (Quantified Monotone Algebraic Circuits)**.** *A quantified monotone algebraic circuit has the form*

$$\mathtt{Q}_{z_1}^{(1)}\mathtt{Q}_{z_2}^{(2)}\cdots\mathtt{Q}_{z_m}^{(m)}\mathcal{C}(\mathbf{x},\mathbf{z})$$

where $|\mathbf{z}| = m$, $\mathtt{Q}^{(i)} \in \{\mathtt{sum},\mathtt{prod}\}$ for each $i \in [m]$, and $\mathcal{C}$ is a monotone algebraic circuit. The size of the quantified monotone algebraic circuit above is $m + \mathrm{size}(\mathcal{C})$. ◊

Quantified monotone circuits of polynomial size can clearly compute any polynomial in mVNP. It is therefore interesting to check if there is any polynomial of *bounded degree* that is outside mVNP. This turns out to be a tricky question. A reason for that is as follows.

Lemma 3.4. *Let $f(\mathbf{x})$ be a monotone polynomial whose support cannot be written as a non-trivial product of two sets. Further for some monotone polynomial $g(\mathbf{x},\mathbf{z})$, suppose $f(\mathbf{x}) = \mathtt{Q}_{z_1}^{(1)}\mathtt{Q}_{z_2}^{(2)}\cdots\mathtt{Q}_{z_m}^{(m)}g(\mathbf{x},\mathbf{z})$ with $\mathtt{Q}^{(i)} \in \{\mathtt{sum},\mathtt{prod}\}$ for each $i \in [m]$.*

Then $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x},\bar{1}))$.

This is an extension of the observation due to Yehudayoff [Yeh19] and essentially shows that any lower bound proof against mVNP for a polynomial computable by quantified monotone circuits will require an argument that relies on some structure in the coefficients.

While there are instances of such arguments in the literature [Yeh19, CDGM22, CDM21], extending those ideas to work against mVNP does not seem like an easy task. The following theorem sheds some light on the cause of this difficulty.

Theorem 3.5. *Suppose $f(\mathbf{x})$ is an n-variate, degree-d polynomial computed by a quantified monotone circuit of size s, which uses ℓ summation gates. Then for a set of variables $\mathbf{w}$ of size at most $d \cdot \ell$, there is a monotone circuit $h(\mathbf{x},\mathbf{w})$ of size at most $d \cdot s$, and a polynomial $A(\mathbf{w})$ such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b}\in\{0,1\}^{|\mathbf{w}|}} A(\mathbf{w})\cdot h(\mathbf{x},\mathbf{w}), \tag{3.6}$$

where $A(\mathbf{w})$ potentially has size and degree that is exponential in n and ℓ.

We now discuss how Theorem 3.5 helps us understand the main barriers towards separating quantified monotone VP from mVNP.

1. If the polynomial $A(\mathbf{w})$ from Theorem 3.5 were to have degree and size that is polynomial in n, then quantified monotone VP would collapse to mVNP. Further since A is free of $\mathbf{x}$, its exponential degree and size can be leveraged only for designing coefficients of f. Moreover, the monotone nature of A and h ensures that $A(\mathbf{1})$ is the largest value, and contributes *equally* to all monomials in the support of f, since $\text{supp}(f) = \text{supp}(h(\mathbf{x}, \mathbf{w} = \mathbf{1}))$.

2. Another consequence that is quite interesting is the following. Suppose there is a different monotone polynomial $B(\mathbf{w})$ of small degree and size that agrees with $A(\mathbf{w})$ on all $\{0,1\}$-inputs, then $f(\mathbf{x}) = \sum_{\mathbf{b}} B(\mathbf{w})h(\mathbf{x}, \mathbf{w})$. That is, we can replace A by B in our expression and then f clearly has an efficient 'mVNP-expression'.

   Thus, any separation between mVNP and quantified monotone VP will provide a polynomial $A(w)$ which is hard to compute for mVNP, even as a function over the boolean hypercube; a result that perhaps stands on its own.

Monotone circuits with summation and production gates Next we consider a model that further generalises quantified monotone circuits. Here summation and production gates are allowed to appear anywhere in the circuit.

Definition 3.7 (Algebraic circuits with summation and production gates)**.** *An algebraic circuit with summation, production gates is a directed acyclic graph with leaves (nodes with in-degree zero) labelled by formal variables and constants from the field, and other nodes labelled by addition* $(+)$*, multiplication* $(\times)$*, summation* $(\texttt{sum}_z)$ *or production* $(\texttt{prod}_z)$*.*

The leaves compute their labels, and addition, multiplication nodes compute the operation they are labelled by, on the polynomials along its incoming edges. The nodes labelled by summation or production computes the polynomial described in Definition 2.9. There is a unique node of out-degree zero called the root, and the circuit is said to compute the polynomial computed at the root.

The size *of a circuit,* $\mathcal{C}$*, denoted by* $\text{size}(\mathcal{C})$*, is the number of nodes in the graph.*

An algebraic circuit with summation, production gates is said to be monotone*, if all the constants appearing in it are non-negative.* ◊

Note that even in the non-monotone setting this model is clearly as powerful as quantified circuits and less powerful than circuits with projection gates. Therefore since Malod [Mal11] showed that quantified circuits and circuits with projection gates are equivalent in power, the class of polynomials efficiently computable by this model is again VPSPACE.

In the monotone setting, however, it is not clear if the power of quantified monotone circuits is the same as that of this model. In particular, it is unclear if a version of Lemma 3.4 is true for

this model. However, we show that even this additional power does not help much in monotone computation of transparent polynomials.

Theorem 3.8. *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial f, has size at least $|\mathrm{supp}(f)|/4$.*

This shows that transparent polynomials with large support are hard even for this model. Recall that one way to refute the τ-conjecture for Newton polygons is to show a transparent polynomial in (non-monotone) VP. Theorem 3.8 shows that any transparent polynomial from VP that refutes the conjecture would also witness a separation between VP and a class potentially much bigger than mVNP[4]. Even though stark separations between monotone and non-monotone models are not unheard of [HY13, CDM21], such a result would be very interesting and would further highlight the power of subtractions.

Monotone circuits with projection gates. Finally, adapting the definition of VPSPACE due to Poizat (Definition 2.7) [Poi08], we define monotone circuits with projection gates.

Definition 3.9 (Monotone algebraic circuits with projection gates)**.** *A* monotone algebraic circuit with projection gates *is an algebraic circuit with projection (as defined in Definition 2.7) in which only non-negative constants from the field are allowed to appear as labels of leaves.*

The size *of a monotone algebraic circuit with projection gates is the number of nodes in the underlying graph.* ◊

This model is clearly at least as powerful as monotone circuits with summation and production gates, since $\mathtt{sum}_z = \mathtt{fix}_{(z=0)} + \mathtt{fix}_{(z=1)}$ and $\mathtt{prod}_z = \mathtt{fix}_{(z=0)} \times \mathtt{fix}_{(z=1)}$. It would therefore be interesting to show a separation between the power of the two models.

Even though we are unable to do that, we show that monotone circuits with projection gates are indeed more powerful than quantified monotone circuits, with a $2^{\Omega(\sqrt{n})}$ separation. We do so by first showing that the Permanent family is efficiently computable by the first model (Theorem 7.1), and then using Lemma 3.4 against the second model.

Theorem 3.10. *The polynomial family $\{\mathsf{Perm}_n\}$ can be computed by a monotone circuit with projection gates of size $O(n^3)$, but any quantified monotone circuit computing it must have size $2^{\Omega(\sqrt{n})}$.*

We end this section with a conjecture. We believe that transparency is a highly restrictive property, especially for monotone computation. Therefore we conjecture that if f is a transparent polynomial being computed by a monotone circuit with projection gates of size s, then $|\mathrm{supp}(f)| \leq 2^{\mathrm{polylog}(s)}$.

[4]That is, the class of bounded degree polynomials computable by monotone algebraic circuits with summation and production gates.

4 Monotone succinct algebraic branching programs

In this section we prove Theorem 3.2.

Theorem 3.2. *If an n-variate polynomial $f(\mathbf{x})$ of degree $\mathrm{poly}(n)$ is computable by a monotone succinct ABP of complexity $\mathrm{poly}(n)$, then $f(\mathbf{x}) \in \mathsf{mVNP}$.*

Proof. Let $\mathcal{A} = (B, \mathbf{s}, \mathbf{t}, \ell)$ be the monotone succinct ABP computing f, with $|\mathbf{s}| = |\mathbf{t}| = r$.

Claim 4.1. *If $\ell > 1$, then $\ell \leq \deg(f) + 2$.*

Proof. Let $b(\mathbf{u}, \mathbf{v}, \mathbf{x})$ be the *monotone* $(2r+n)$-variate polynomial computed by the circuit B. Due to the monotonicity of B, for any $\mathbf{e} \in \mathbb{N}^n$ we have that if the monomial $\mathbf{x}^{\mathbf{e}}$ appears in any edge-label $(\mathbf{a}, \mathbf{b})$, then it also appears in the label of $(\bar{1}, \bar{1})$. Therefore $\deg_{\mathbf{x}}(B(\mathbf{a}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(B(\bar{1}, \bar{1}, \mathbf{x}))$ for all $\mathbf{a}, \mathbf{b}$. Similarly, $\deg_{\mathbf{x}}(B(\mathbf{s}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(B(\mathbf{s}, \bar{1}, \mathbf{x}))$ and $\deg_{\mathbf{x}}(B(\mathbf{a}, \mathbf{t}, \mathbf{x})) \leq \deg_{\mathbf{x}}(B(\bar{1}, \mathbf{t}, \mathbf{x}))$ for all $\mathbf{a}, \mathbf{b}$. This shows that if $\ell > 1$, then

$$\deg(f) = \deg(B(\mathbf{s}, \bar{1}, \mathbf{x}) \cdot B(\bar{1}, \bar{1}, \mathbf{x})^{\ell-2} \cdot B(\bar{1}, \mathbf{t}, \mathbf{x})) \geq \ell - 2. \qquad \square$$

As a result of the above claim, for $d = \deg(f)$, we have the following.

$$\begin{aligned}
f(\mathbf{x}) &= B(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d-1} (\text{sum of } \mathbf{s}\text{–}\mathbf{t} \text{ paths through } j \text{ intermediate vertices}) \\
&= B(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d-1} \left(\sum_{\mathbf{a}_1, \ldots, \mathbf{a}_j \in \{0,1\}^r} B(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \cdot \left(\prod_{k=1}^{j-1} B(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \cdot B(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right) \\
&= B(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{\mathbf{a}_1, \ldots, \mathbf{a}_{d-1} \in \{0,1\}^r} \sum_{j=1}^{d-1} 2^{-r(d-1-j)} \left(B(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \cdot \left(\prod_{k=1}^{j-1} B(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \cdot B(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right) \\
&= \sum_{\mathbf{a}_1, \ldots, \mathbf{a}_{d-1}} \left(2^{-r(d-1)} B(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d-1} 2^{-r(d-1-j)} B(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \left(\prod_{k=1}^{j-1} B(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) B(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right)
\end{aligned}$$

which is clearly a monotone VNP expression, since $d = \mathrm{poly}(n)$ and B is a monotone circuit of size $\mathrm{poly}(n)$. $\square$

5 Quantified monotone circuits

In this section we first prove Lemma 3.4, which is an extension of the following observation due to Yehudayoff [Yeh19].

Observation 5.1 ([Yeh19])**.** *Let $g(\mathbf{x}, z)$ be a monotone polynomial and let $c > 0$. Then for any monomial $m = \mathbf{x}^{\mathbf{e}} z^j$ in the support of g, $\mathbf{x}^{\mathbf{e}} \in \mathrm{supp}(g, z = c)$.*

We now restate Lemma 3.4 and complete its proof.

Lemma 3.4. *Let $f(\mathbf{x})$ be a monotone polynomial whose support cannot be written as a non-trivial product of two sets. Further for some monotone polynomial $g(\mathbf{x},\mathbf{z})$, suppose $f(\mathbf{x}) = \mathtt{Q}^{(1)}_{z_1}\mathtt{Q}^{(2)}_{z_2}\cdots\mathtt{Q}^{(m)}_{z_m} g(\mathbf{x},\mathbf{z})$ with $\mathtt{Q}^{(i)} \in \{\mathtt{sum},\mathtt{prod}\}$ for each $i \in [m]$.*

Then $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x},\bar{1}))$.

Proof. Observe that it is enough to show the statement of the lemma for $m = 1$.

Therefore, suppose $f(\mathbf{x}) = \mathtt{sum}_z g(\mathbf{x},z)$, then $f(\mathbf{x}) = g(\mathbf{x},0) + g(\mathbf{x},1)$, and hence $\mathrm{supp}(f) = \mathrm{supp}(g(\mathbf{x},1))$, since g is monotone.

Next, $f(\mathbf{x}) = \prod_z g(\mathbf{x},z)$ means that $f(\mathbf{x}) = g(\mathbf{x},0)\cdot g(\mathbf{x},1)$. As $\mathrm{supp}(f)$ cannot be written as a non-trivial product of two sets, and since g is monotone, this must mean that $g(\mathbf{x},0)$ is a constant and $\mathrm{supp}(f(\mathbf{x})) = \mathrm{supp}(g(\mathbf{x},1))$ as claimed. □

Let us now move on to the proof of Theorem 3.5, which we first restate.

Theorem 3.5. *Suppose $f(\mathbf{x})$ is an n-variate, degree-d polynomial computed by a quantified monotone circuit of size s, which uses ℓ summation gates. Then for a set of variables $\mathbf{w}$ of size at most $d\cdot\ell$, there is a monotone circuit $h(\mathbf{x},\mathbf{w})$ of size at most $d\cdot s$, and a polynomial $A(\mathbf{w})$ such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b}\in\{0,1\}^{|\mathbf{w}|}} A(\mathbf{w})\cdot h(\mathbf{x},\mathbf{w}), \tag{3.5}$$

where $A(\mathbf{w})$ potentially has size and degree that is exponential in n and ℓ.

The proof requires us to repeatedly use the following simple observation. It is easy to verify and therefore we omit its proof.

Observation 5.2 (Product of exponential sums)**.**

$$\mathtt{prod}_z \mathtt{sum}_{\mathbf{y}} g(\mathbf{x},\mathbf{y},z) = \mathtt{sum}_{\mathbf{y}_0,\mathbf{y}_1}\left(g(\mathbf{x},\mathbf{y}_0,0)\cdot g(\mathbf{x},\mathbf{y}_1,1)\right)$$

We now use a toy example to exhibit the trivial way of moving from a quantified expression to an exponential sum, using Observation 5.2.

$$\begin{aligned}
f(x) &= \mathtt{sum}_{y_1}\mathtt{prod}_{z_1}\mathtt{sum}_{y_2}\mathtt{prod}_{z_2,z_3}\mathtt{sum}_{y_3} g(x,y_1,y_2,y_3,z_1,z_2,z_3)\\
&= \mathtt{sum}_{y_1}\mathtt{prod}_{z_1}\mathtt{sum}_{y_2}\mathtt{prod}_{z_2}\mathtt{sum}_{y_{3,0},y_{3,1}}\left(\prod_{a_3\in\{0,1\}} g(x,y_1,y_2,y_{3,a_3},z_1,z_2,a_3)\right)\\
&= \mathtt{sum}_{y_1}\mathtt{prod}_{z_1}\mathtt{sum}_{y_2,y_{3,(00)},y_{3,(01)},y_{3,(10)},y_{3,(11)}}\left(\prod_{a_2,a_3\in\{0,1\}} g(\ldots,y_{3,(a_2a_3)},z_1,a_2,a_3)\right)\\
&= \mathtt{sum}_{y_1}\mathtt{sum}_{y_{2,*},y_{3,***}}\left(\prod_{a_1,a_2,a_3\in\{0,1\}} g(x,y_1,y_{2,a_1},y_{3,(a_1a_2a_3)},a_1,a_2,a_3)\right)
\end{aligned}$$

In the last line, $*$ runs over $\{0,1\}$, so there are $1+2+8=11$ auxiliary variables in total. Note that y_3 has 8 copies, which is due to the 3 production gates 'above' the summation gate labelled by it. Similarly y_2 has just 2 copies, while y_1 has just one. In particular, it should be noted that the number of such copies is independent of the number of *alternations*. Also if instead of single auxiliary variables y_2 and y_3 we had sets of auxiliary variables $\mathbf{y}_2$ and $\mathbf{y}_3$, nothing much would change. That is, we would have had 8 copies of the set $\mathbf{y}_3$ and 2 copies of $\mathbf{y}_2$, irrespective of their sizes.

In general, what this shows is that we can trivially move from a quantified expression to an expression which has the form

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{Y}} \prod_{\mathbf{a}\in\{0,1\}^r} g_{\mathbf{a}}(\mathbf{x}, \mathbf{y}_{\mathbf{a}})$$

where $\mathbf{Y} = \cup_{\mathbf{a}} \{\mathbf{y}_{\mathbf{a}}\}$, r is the number of production gates in the quantified expression, $|\mathbf{Y}|$ is potentially exponential (since the number of copies of some auxiliary variable might be exponential) but $g_{\mathbf{a}}(\mathbf{x}, \mathbf{y}_{\mathbf{a}}) = g(\mathbf{x}, \mathbf{y} = \mathbf{y}_{\mathbf{a}}, \mathbf{z} = \mathbf{a})$ for a poly-sized circuit $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

The key observation allowing us to prove Theorem 3.5 is that if f has degree d, then the number of copies of each auxiliary variable needed in the outer summation gate is at most d. This is because, due to monotonicity, $\deg_{\mathbf{x}}(g_{\mathbf{a}}(\mathbf{x}, \mathbf{y}_{\mathbf{a}})) \neq 0$ for only d many $\mathbf{a} \in \{0,1\}^r$.

Moving on to a formal proof, we introduce a new shorthand for the remainder of this section. For a vector $\mathbf{a} = \{a_1, a_2, \ldots, a_\ell\}$ and a number $k \leq \ell$, we use $\mathbf{a}[:k]$ to denote the *prefix* vector $\{a_1, a_2, \ldots, a_k\}$. With this new notation, we can express the last line of our toy example as follows.

$$f(x) = \mathtt{sum}_{y_1} \mathtt{sum}_{y_{2,*}, y_{3,***}} \left(\prod_{\mathbf{a}\in\{0,1\}^3} g(x, y_1, y_{2,\mathbf{a}[:1]}, y_{3,\mathbf{a}[:3]}, a_1, a_2, a_3) \right)$$

We are now ready to prove Theorem 3.5. We start by recalling the statement of the theorem.

Theorem 3.5. *Suppose $f(\mathbf{x})$ is an n-variate, degree-d polynomial computed by a quantified monotone circuit of size s, which uses ℓ summation gates. Then for a set of variables $\mathbf{w}$ of size at most $d \cdot \ell$, there is a monotone circuit $h(\mathbf{x}, \mathbf{w})$ of size at most $d \cdot s$, and a polynomial $A(\mathbf{w})$ such that,*

$$f(\mathbf{x}) = \sum_{\mathbf{b}\in\{0,1\}^{|\mathbf{w}|}} A(\mathbf{w}) \cdot h(\mathbf{x}, \mathbf{w}), \tag{3.5}$$

where $A(\mathbf{w})$ potentially has size and degree that is exponential in n and ℓ.

Proof. The first step is to obtain a trivial exponential sum for the quantified expression, as in the discussion above.

Claim 5.3. *Suppose $f(\mathbf{x})$ can be expressed as the following quantified circuit.*

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{y}_1}\mathtt{prod}_{\mathbf{z}_1}\mathtt{sum}_{\mathbf{y}_2}\mathtt{prod}_{\mathbf{z}_2}\cdots\mathtt{prod}_{\mathbf{z}_k}\mathtt{sum}_{\mathbf{y}_{k+1}} g(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_{k+1}, \mathbf{z}_1, \ldots, \mathbf{z}_k)$$

Let $m_i = |\mathbf{z}_i|$, and further let $M_i = m_1 + m_2 + \cdots + m_i$, for each $i \in [k]$. Also, let $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2 \cup \cdots \cup \mathbf{y}_{k+1}$, and $\mathbf{z} = \mathbf{z}_1 \cup \mathbf{z}_2 \cup \cdots \cup \mathbf{z}_k$

Then $f(\mathbf{x})$ can also be expressed as the following exponential sum.

$$f(\mathbf{x}) = \mathtt{sum}_{\mathbf{Y}}\left(\prod_{\mathbf{a}\in\{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \mathbf{y}_{3,\mathbf{a}[:M_2]}, \ldots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]}, \mathbf{z} = \mathbf{a})\right)$$

Here $\mathbf{Y}$ is a set of all y-variables, of size $\left(1 + \sum_i 2^{M_i}\right)$ that is defined as follows.

$$\mathbf{Y} = \bigcup_{\mathbf{a}\in\{0,1\}^{M_k}} (\mathbf{y}_1 \cup \mathbf{y}_{2,\mathbf{a}[:M1]} \cup \cdots \cup \mathbf{y}_{k+1,\mathbf{a}[:M_k]}) \qquad \square$$

Even though the claim is fairly verbose, it is easy to verify given the discussion before the lemma, so we will not explicitly prove it.

As the next step, we shall use the fact that the 'inner circuit' g is monotone, to lower bound the degree of f.

$$\begin{aligned}
\deg(f) &= \deg_{\mathbf{x}}\left(\mathtt{sum}_{\mathbf{Y}}\left(\prod_{\mathbf{a}\in\{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \ldots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]}, \mathbf{z} = \mathbf{a})\right)\right) \\
(g \text{ is monotone}) &= \deg_{\mathbf{x}}\left(\prod_{\mathbf{a}\in\{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a})\right) \\
&\geq \sum_{\mathbf{a}\in\{0,1\}^{M_k}} \deg(g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a}))
\end{aligned}$$

Therefore, since f has degree $dg(\mathbf{x}, \mathbf{y}, \mathbf{a})$, it must be the case that for all but d fixings $\mathbf{a}$ of $\mathbf{z}$, $g(\mathbf{x}, \mathbf{y}, \mathbf{a})$ is a constant in terms of $\mathbf{x}$ for any $\{0,1\}$-assignment[5] to the variables in $\mathbf{y}$.

Let $\mathcal{A} := \left\{\mathbf{a} \in \{0,1\}^{M_k} : \deg_{\mathbf{x}}(g(\mathbf{x}, \mathbf{b}, \mathbf{a})) > 0 \text{ for some } \mathbf{b} \in \{0,1\}^{|\mathbf{y}|}\right\}$, and let $\mathcal{A}_0 := \{0,1\}^{M_k} \setminus \mathcal{A}$. We therefore have that $|\mathcal{A}| \leq d$. Further, let $\mathbf{Y}_1 := \bigcup_{\mathbf{a}\in\mathcal{A}}(\mathbf{y}_1 \cup \mathbf{y}_{2,\mathbf{a}[:M1]} \cup \cdots \cup \mathbf{y}_{k+1,\mathbf{a}[:M_k]})$, and let $\mathbf{Y}_0 := \mathbf{Y} \setminus \mathbf{Y}_1$. Note that now $|\mathbf{Y}_1| \leq |\mathcal{A}| \cdot |\mathbf{y}| \leq d \cdot m$.

We can now simplify the exponential sum in Claim 5.3 and finish the proof as follows, where

[5]Note that $g(\mathbf{x}, \mathbf{y}, \mathbf{a})$ can be a non-constant polynomial in $\mathbf{x}$ and still have this property: e.g. $y + x(y^2 - y)$. It can be checked that the proof goes through despite this.

$\mathbf{y_a}$ refers to $(\mathbf{y}_1, \mathbf{y}_{2,\mathbf{a}[:M_1]}, \cdots, \mathbf{y}_{k+1,\mathbf{a}[:M_k]})$.

$$
\begin{aligned}
f(\mathbf{x}) &= \mathtt{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \\
\text{(for appropriate } \mathbf{y_a}) &= \mathtt{sum}_{\mathbf{Y}} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
\text{(first term ``}\mathbf{x}\text{-free'')} &= \mathtt{sum}_{\mathbf{Y}} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
&= \mathtt{sum}_{\mathbf{Y}_1, \mathbf{Y}_0} \left(\left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \\
\text{(regroup terms)} &= \mathtt{sum}_{\mathbf{Y}_1} \left(\mathtt{sum}_{\mathbf{Y}_0} \left(\prod_{\mathbf{a} \in \mathcal{A}_0} g(\mathbf{0}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \right) \cdot \left(\prod_{\mathbf{a} \in \mathcal{A}} g(\mathbf{x}, \mathbf{y_a}, \mathbf{z} = \mathbf{a}) \right) \\
\text{(simplify)} &= \mathtt{sum}_{\mathbf{Y}_1} A(\mathbf{Y}_1) \cdot h(\mathbf{x}, \mathbf{Y}_1)
\end{aligned}
$$

As claimed, the size of h is at most $|\mathcal{A}| \cdot \text{size}(g) \leq d \cdot s$, while $A(\mathbf{Y}_1)$ is a fairly structured polynomial despite its exponential size and degree.

6 Monotone circuits with summation and production gates

In this section, we prove Theorem 3.8. We start by recalling the theorem.

Theorem 3.8. *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial f, has size at least $|\text{supp}(f)|/4$.*

This result is an extension of the ideas in the work of Hrubeš and Yehudayoff [HY21]. Their argument shows that any bivariate monotone circuit of size s that computes a polynomial with *convexly independent support* outputs a polynomial with support at most $4s$. They achieve this by keeping track of the largest polygon (having most vertices) that one can build using the polynomials computed at all the gates in the circuit. They then inductively show that no gate (leaf, addition, multiplication) can increase the number of vertices by 4. We are able to show the same bound for production and summation gates, by working with a monotone bivariate circuit over y_1, y_2 that is allowed some auxiliary variables z for summations and productions.

An important component of the proof in [HY21] is that if the sum or product of two monotone polynomials in convexly independent, then so are each of the two inputs. However, the allowing for summations and productions means that some monomials that are computed internally could get "zeroed out". In fact, summation and production gates do not quite "preserve convex dependencies". For example, the convexly dependent support $\{y_1y_2, y_1y_2z, y_1y_2z^2\}$ when passed through $\mathtt{sum}_z$ produces just $\{y_1y_2\}$, which is convexly independent.

In order to prove Theorem 3.8, we get around this by working directly with the support projected down to the "true" variables, which we call **y**-support in our arguments. It turns out that summations and productions indeed preserve convex dependencies that are in the **y** support of the input polynomial. We now show a complete proof.

We start by recalling the concepts of *shadow complexity* and *transparent polynomials*.

Definition 6.1 (Shadow complexity [HY21])**.** *For a polynomial $f(x_1, \ldots, x_n)$, its shadow complexity $\sigma(f)$ is defined as follows.*

$$\sigma(f) := \max_{L:\mathbb{R}^n \to \mathbb{R}^2} |\mathsf{vert}(L(\mathsf{Newt}(f)))| \qquad \Diamond$$

For any n, a set of points in $\mathbb{R}^n$ is said to be *convexly independent* if no point in the set can be written as a convex combination of other points from the set. Note that if a polynomial has *convexly independent support*, then all the monomials in its support correspond to vertices of its Newton polytope. The following definition is an even stronger condition.

Definition 6.2 (Transparent polynomials [HY21])**.** *A polynomial f is said to be transparent, if $\sigma(f) = |\mathrm{supp}(f)|$.* $\Diamond$

The following lemma states that the linear map that witnesses the shadow complexity of a polynomial over the reals, can be assumed to be "integral" without loss of generality.

Lemma 6.3 (Consequence of [HY21, Lemma 4.2])**.** *Let $f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ be an n-variate polynomial. Then there is an $M \in \mathbb{Z}^{2\times n}$, such that for $L(\mathbf{e}) := M \cdot \mathbf{e}$, $|\mathsf{vert}(L(\mathsf{Newt}(f)))| = \sigma(f)$.*

We also require the following concepts from the work of Hrubešand Yehudayoff [HY21].

Definition 6.4 (Laurent polynomials and high powered circuits)**.** *A* Laurent polynomial *over the variables $\{x_1, \ldots, x_n\}$ and a field $\mathbb{F}$, is a finite $\mathbb{F}$-linear combination of terms of the form $x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$, where $p_1, p_2, \ldots, p_n \in \mathbb{Z}$.*

A high powered circuit over the variables $\{x_1, \ldots, x_n\}$ and a field $\mathbb{F}$, is an algebraic circuit whose leaves can compute terms like $\alpha x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}$ for any $\alpha \in \mathbb{F}$ and $\mathbf{p} \in \mathbb{Z}^n$. In other words, a high powered circuit can compute an arbitrary Laurent monomial with size 1; the size of the high powered circuit is the total number of nodes as usual. $\Diamond$

Using the above definition, we can easily infer the following by replacing each leaf with the corresponding Laurent monomial.

Observation 6.5. *Let $f(\mathbf{x})$ be computable by a monotone circuit of size s, and suppose $\sigma(f) = k$. Then there exists a bivariate Laurent polynomial $P(y_1, y_2)$ that is computable by a high powered circuit of size s, whose Newton polygon has k vertices.*

We now have all the concepts required to prove the main theorem of this section, Theorem 3.8. The following results and their proofs closely follow those in [HY21]. We reproduce the overlapping parts for the sake of completeness and ease of exposition.

Lemma 6.6 ([HY21, Lemma 5.8]). *Let $A, B \subset R^2$ be finite sets, such that $A + B$ is convexly independent. Then if $|A| \geq |B|$, then either $|A|, |B| \leq 2$ or $|B| = 1$.*

Theorem 6.7 (Extension of [HY21, Theorem 5.9]). *Let $f(y_1, y_2)$ be a monotone Laurent polynomial with convexly independent support, and let $C(y_1, y_2, \mathbf{z})$ be a monotone high-powered circuit with summation and production gates[6], that computes f. Then* $\text{size}(C) \geq |\text{supp}(f)|/4$.

Proof. For a multi-set[7] $\mathcal{A}$ that contains sets of points in $\mathbb{R}^2$, we define a measure μ that relates to the "largest" convexly independent set that can be constructed using it. For a sub-collection $\mathcal{B} \subseteq \mathcal{A}$ and a map $v : \mathcal{B} \to \mathbb{R}^2$, the resulting set $\mathcal{B}(v)$ is defined as follows.

$$\mathcal{B}(v) := \bigcup_{A \in \mathcal{B}} (\{v(A)\} + A)$$

The measure μ is then defined as follows.

$$\mu(\mathcal{A}) := \max_{\mathcal{B}, v} \{|\mathcal{B}(v)| : \mathcal{B}(v) \text{ is convexly independent}\} \tag{6.8}$$

For a Laurent polynomial $g(y_1, y_2, \mathbf{z})$, let $\text{supp}_{\mathbf{y}}(g) := \{(a, b) : \exists \mathbf{e}, y_1^a y_2^b \mathbf{z}^{\mathbf{e}} \in \text{supp}(g)\}$ be its $\mathbf{y}$-support. Corresponding to the circuit $C(y_1, y_2, \mathbf{z})$ of size s, we will consider the collection $\mathcal{A}$ of s sets, which will be the $\mathbf{y}$-supports of the polynomials computed by the s gates. The following claim will help us prove the theorem by induction.

Claim 6.9. *For $\mathcal{A}' = \mathcal{A} \cup \{B\}$, and $A_1, A_2 \in \mathcal{A}$,*

$$\mu(\mathcal{A}') \leq \mu(\mathcal{A}) + |B|, \tag{6.10}$$
$$\mu(\mathcal{A}') \leq \mu(\mathcal{A}) + 2 \qquad \text{if } B = u + A_1, \tag{6.11}$$
$$\mu(\mathcal{A}') \leq \mu(\mathcal{A}) + 4 \qquad \text{if } B = A_1 \cup A_2, \tag{6.12}$$
$$\mu(\mathcal{A}') \leq \mu(\mathcal{A}) + 4 \qquad \text{if } B = A_1 + A_2, \tag{6.13}$$
$$\mu(\mathcal{A}') \leq \mu(\mathcal{A}) + 4 \qquad \text{if } B = A_1 + A' \text{ for } A' \subseteq A_1\,. \tag{6.14}$$

Proof. It is trivial to see that (6.10) holds.

For (6.11), suppose $\mathcal{B}$ is the subset that achieves $\mu(\mathcal{A}') > \mu(\mathcal{A})$. Then $A_1, B \in \mathcal{B}$ as otherwise one can mimic the contribution of B using A_1; further $v(A_1) \neq v(B) + u$ because otherwise the translates of A_1 and B overlap. Now note that $(\{v(A_1)\} + A_1) \cup (\{v(B)\} + B)$ is a convexly independent set of points, and also that $(\{v(A_1)\} + A_1) \cup (\{v(B)\} + B) = \{v(A_1), v(B) + u\} + A_1$. Therefore by Lemma 6.6, we see that $|B| = |A_1| \leq 2$, which finises the proof using (6.10).

[6] All auxiliary variables only appear with non-negative powers in the circuit.

[7] We assume that copies of the same set $A \in \mathcal{A}$ can be referred distinctly.

For (6.12), observe that $\mu(\mathcal{A}) \leq \mu(\mathcal{A} \cup A_1, A_2)$. The required bound then follows by two applications of (6.11).

In (6.13), if B is convexly *dependent*, then it cannot contribute to $\mu(\mathcal{A}')$, so suppose it is. Assuming $|A_1| \geq |A_2|$ without loss of generality, by Lemma 6.6, either $|B| \leq |A_1| \cdot |A_2| \leq 4$, or $B = u + A_1$ for some u, and (6.11) finishes the proof.

Clearly (6.13) implies (6.14), as its proof does not depend on whether $A_2 \in \mathcal{A}$, or $A_2 \not\subseteq A_1$. □

We now argue that the polynomial computed at every gate in $C(y_1, y_2, \mathbf{z})$ has convexly independent $\mathbf{y}$-support. Since the $\mathbf{y}$-supports of addition and multiplication gates are unions and Minkowski sums of their children respectively, if any of their input is convexly dependent, then so is the output. For a summation gate $g = \mathtt{sum}_z g'$, $\mathrm{supp}_{\mathbf{y}}(g) = \mathrm{supp}_{\mathbf{y}}(g')$ using Lemma 3.4. For a production gate $g = \mathtt{prod}_z g'$, $\mathrm{supp}_{\mathbf{y}}(g) = S' + \mathrm{supp}_{\mathbf{y}}(g')$ for some $S' \subseteq \mathrm{supp}_{\mathbf{y}}(g')$, so any convex dependency in $\mathrm{supp}_{\mathbf{y}}(g')$ would transfer to $\mathrm{supp}_{\mathbf{y}}(g)$. Since the output of $C(x, y, \mathbf{z})$ is convexly independent, the above observations imply that each gate $g \in C$ has convexly independent $\mathrm{supp}_{\mathbf{y}}(g)$.

Let us now prove the theorem by inductively building the collection $\mathcal{A}$ with respect to the circuit C: a gate is added only after adding all of its children. When the gate being added is a leaf, then μ increases by at most 1 due to (6.10). For an addition gate computing g, $\mathrm{supp}_{\mathbf{y}}(g)$ is the union of the (x, y)-supports of its children; so we can apply (6.12). For an multiplication gate computing g, $\mathrm{supp}_{\mathbf{y}}(g)$ is the Minkowski sum of the (x, y)-supports of its children; so we can use (6.13). For a summation gate that computes g, note that its (x, y)-support is exactly the same as that of its child ((5.1)); therefore (6.11) applies. Finally for a production gate, we can use (6.14), as $\mathrm{supp}_{\mathbf{y}}(\mathtt{prod}_z g) = \mathrm{supp}_{\mathbf{y}}(g|_{z=0}) + \mathrm{supp}_{\mathbf{y}}(g|_{z=1})$, and $\mathrm{supp}_{\mathbf{y}}(g|_{z=0}) \subseteq \mathrm{supp}_{\mathbf{y}}(g|_{z=1}) = \mathrm{supp}_{\mathbf{y}}(g)$.

Since the measure μ increases by at most 4 in each of the s steps, we have that $|\mathrm{supp}(f)| \leq \mu(\mathcal{A}) \leq 4s$, as required. □

The above result then lets us prove Theorem 3.8, which we first restate.

Theorem 3.8. *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial f, has size at least $|\mathrm{supp}(f)|/4$.*

Proof. Let C be a quantified monotone circuit computing f_n, of size s. Since $f_n(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ is transparent, there exists a matrix $M \in \mathbb{Z}^{2\times n}$, such that the linear map $L(\mathbf{e}) = M\mathbf{e}$, satisfies $|\mathrm{vert}(L(\mathsf{Newt}(f)))| = |\mathrm{supp}(f)|$. Further using Observation 6.5, there exists a size-s high powered monotone circuit with summation and production gates, that computes a Laurent polynomial $P(y_1, y_2)$ which has $|\mathrm{supp}(f)|$ vertices in its Newton polytope. The bound then easily follows from Theorem 6.7. □

7 Monotone circuits with projection gates

In this section, we prove Theorem 3.10. First we describe an efficient monotone circuit with projection gates that computes Perm_n.

Theorem 7.1. *There is a monotone circuit with projection gates of size $O(n^3)$ that computes Perm_n.*

Proof. We first define a polynomial P_0 such that all its monomials contain at most one $\mathbf{x}$-variable from each row.

$$\text{Let } P_0(\mathbf{x},\mathbf{y}) := \left(\sum_{j=1}^{n} y_{1,j}x_{1,j}\right)\left(\sum_{j=1}^{n} y_{2,j}x_{2,j}\right)\cdots\left(\sum_{j=1}^{n} y_{n,j}x_{n,j}\right).$$

Note that P_0 has n^2-many auxiliary variables $\mathbf{y}$, one attached to each 'true' variable $x_{i,j}$. We now want to use these to progressively prune the monomials that pick up multiple variables from the jth column by projecting the n variables $y_{1,j},\ldots,y_{n,j}$.

Let $e_1,\ldots,e_n \in \{0,1\}^n$ such that $e_i(k) = 1 \Leftrightarrow i = k$, and define for each $j \in [n]$,

$$P_j := \sum_{i\in[n]} \mathtt{fix}_{(y_{1,j}=e_i(1))}\left(\mathtt{fix}_{(y_{2,j}=e_i(2))}\left(\cdots\left(\mathtt{fix}_{(y_{n,j}=e_i(n))}\left(P_{j-1}\right)\right)\right)\right). \tag{7.2}$$

The following claim is now easy to verify.

Claim 7.3. *For all $j \in [n]$, P_j contains all the monomials from P_{j-1} that are supported on exactly one $\mathbf{x}$-variable from the jth column.*

As a result, the monomials in P_n are exactly those of the monomials in Perm_n. Additionally for each j, the auxiliary variables in P_j are only from the columns $j+1,\ldots,n$; thus $P_n = \mathsf{Perm}_n$.

The size of our circuit is $O(n^3)$, since $\mathrm{size}(P_0) = O(n^2)$ and $\mathrm{size}(P_j) = \mathrm{size}(P_{j-1}) + O(n^2)$. This proves Theorem 7.1. □

Remark 7.4. *Our upper bound above also implies that any polynomial (family) that can be expressed as the permanent of a monotone matrix of size $\mathrm{poly}(n)$ (called monotone p-projection of Perm_n) can also be computed by efficient monotone circuits with projection gates. Although Perm_n is complete for non-monotone* VNP*, it is* not *the case that all monotone polynomials in* VNP *are monotone p-projections of Perm_n, as shown by Grochow [Gro17].* ◊

Finally, we complete the proof of Theorem 3.10.

Theorem 3.10. *The polynomial family $\{\mathsf{Perm}_n\}$ can be computed by a monotone circuit with projection gates of size $O(n^3)$, but any quantified monotone circuit computing it must have size $2^{\Omega(\sqrt{n})}$.*

Proof. Trivially follows from Theorem 7.1 and Lemma 3.4, since Perm_n is irreducible. □

8 Conclusion

Our work is an attempt at understanding the hardness of transparent polynomials for monotone algebraic models. We observe that the lower bound of Hrubeš and Yehudayoff [HY21] extends beyond monotone VNP, and therefore turn to exploring the class VPSPACE from the non-monotone world. This exploration reveals that the natural monotone analogues of the multiple equivalent definitions of VPSPACE have contrasting powers. Additionally, transparent polynomials turn out to be as hard for some of these analogues as they are for usual monotone circuits. Following are some interesting open threads from our work.

- The first and most natural question related to the motivation behind our work is to prove (or refute) our conjecture that a transparent polynomial computed by a size-s monotone circuit with projection gates, has sparsity at most $\exp(\text{poly}\log s)$.

  An immediate hurdle in extending Theorem 3.8 to mVPSPACE is that unlike summations and productions, 0-projections do not preserve convex dependencies, that is, the 0-projection of a convexly dependent polynomial could be convexly independent.

- Along similar lines, a possibly simpler goal is to show a non-monotone circuit upper bound for a transparent polynomial. Note that transparency only restricts the support of the polynomial, so one is free to choose any real coefficients that do not affect the transparency. It could therefore be possible to compute transparent polynomials that have positive and negative coefficients (Lemma 6.3 works for all real polynomials) in VP using fundamentally non-monotone tricks like interpolation. Among other things, such a result would refute the notoriously open τ-conjecture for Newton polygons.

- Another question we would like to highlight is separating mVNP and quantified monotone circuits. As mentioned in the discussion following Theorem 3.5, such a separation would yield a (high degree) polynomial that is hard for mVNP even as a function over the boolean hypercube. Such a polynomial might be of interest, perhaps, even in the non-monotone setting.

References

[CDGM22] Arkadev Chattopadhyay, Rajit Datta, Utsab Ghosal, and Partha Mukhopadhyay. Monotone Complexity of Spanning Tree Polynomial Re-Visited. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 39:1–39:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[CDM21] Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Lower bounds for monotone arithmetic circuits via communication complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 786–799. ACM, 2021.

[CGM22] Arkadev Chattopadhyay, Utsab Ghosal, and Partha Mukhopadhyay. Robustly Separating the Arithmetic Monotone Hierarchy Via Graph Inner-Product. *Electron. Colloquium Comput. Complex.*, TR22-071, 2022. Pre-print available at `arXiv:TR22-071`.

[CKR20] Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman. Monotone Circuit Lower Bounds from Robust Sunflowers. In *LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings*, volume 12118 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2020.

[Gas87] S.B. Gashkov. The complexity of monotone computations of polynomials. *Moscow University Math Bulletin*, (5):1–8, 1987.

[Gro17] Joshua A. Grochow. Monotone Projection Lower Bounds from Extended Formulation Lower Bounds. *Theory of Computing*, 13(18):1–15, 2017.

[GS12] S. B. Gashkov and I. S. Sergeev. A method for deriving lower bounds for the complexity of monotone arithmetic circuits computing real polynomials. *Sbornik. Mathematics*, 203(10), 2012.

[HY13] Pavel Hrubeš and Amir Yehudayoff. Formulas are Exponentially Stronger than Monotone Circuits in Non-commutative Setting. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 10–14. IEEE Computer Society, 2013.

[HY16] Pavel Hrubes and Amir Yehudayoff. On Isoperimetric Profiles and Computational Complexity. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 89:1–89:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[HY21] Pavel Hrubeš and Amir Yehudayoff. Shadows of Newton Polytopes. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[JS82] Mark Jerrum and Marc Snir. Some Exact Complexity Results for Straight-Line Computations over Semirings. *Journal of the ACM*, 29(3):874–897, 1982.

[KP07] Pascal Koiran and Sylvain Perifel. VPSPACE and a Transfer Theorem over the Complex Field. In *Proceedings of the 32nd International Conference on Mathematical Foundations of Computer Science*, MFCS'07, page 359–370, Berlin, Heidelberg, 2007. Springer-Verlag.

[KP09] Pascal Koiran and Sylvain Perifel. VPSPACE and a Transfer Theorem over the Reals. *Computational Complexity*, 18(4):551–575, 2009.

[KPTT15] Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A τ-Conjecture for Newton Polygons. *Foundations of Computational Mathematics*, 15:185–197, 2015.

[Kuz85] SE Kuznetsov. Monotone computations of polynomials and schemes without null-chains. In *Proc. of*, pages 108–109, 1985.

[KZ86] O. M. Kasim-Zade. Arithmetic complexity of monotone polynomials. *Theoretical Problems in Cybernetics. Abstracts of lectures*, page 68–69, 1986.

[Mal11] Guillaume Malod. Succinct Algebraic Branching Programs Characterizing Non-uniform Complexity Classes. In *Fundamentals of Computation Theory - 18th International Symposium, FCT 2011, Oslo, Norway, August 22-25, 2011. Proceedings*, pages 205–216, 2011.

[MR13] Meena Mahajan and B. V. Raghavendra Rao. Small Space Analogues of Valiant's Classes and the Limitations of Skew Formulas. *Computational Complexity*, 22(1):1–38, 2013.

[Poi08] Bruno Poizat. A la recherche de la definition de la complexite d'espace pour le calcul des polynomes a la maniere de Valiant. *J. Symb. Log.*, 73(4):1179–1201, 2008.

[RY11] Ran Raz and Amir Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *Journal of Computer and System Sciences*, 77(1):167–190, 2011.

[Sch76] Claus-Peter Schnorr. A Lower Bound on the Number of Additions in Monotone Computations. *Theor. Comput. Sci.*, 2(3):305–315, 1976.

[Sri19] Srikanth Srinivasan. Strongly Exponential Separation Between Monotone VP and Monotone VNP. *CoRR*, abs/1903.01630, 2019. Pre-print available at `arXiv:1903.01630`.

[SS77] Eli Shamir and Marc Snir. *Lower bounds on the number of multiplications and the number of additions in monotone computations*. IBM Thomas J. Watson Research Division, 1977.

[SS80] Eli Shamir and Marc Snir. On the Depth Complexity of Formulas. *Math. Syst. Theory*, 13:301–322, 1980.

[Val80] Leslie G. Valiant. Negation can be Exponentially Powerful. *Theor. Comput. Sci.*, 12:303–314, 1980.

[Yeh19] Amir Yehudayoff. Separating monotone VP and VNP. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 425–429. ACM, 2019.

A Definitions of VPSPACE relying on boolean computation

In this section we briefly address why we did not study monotone analogues of the definitions due to Koiran and Perifel [KP07, KP09], and Mahajan and Rao [MR13].

Koiran and Perifel define uniform VPSPACE as the class of families $\{f_n\}$ of $\text{poly}(n)$-variate polynomials of degree at most $2^{\text{poly}(n)}$, such that there is a PSPACE machine that computes the *coefficient function* of $\{f_n\}$. Here, the coefficient function of $\{f_n\}$ can be seen to map a pair $(1^n, \mathbf{e})$ to the coefficient of $\mathbf{x}^{\mathbf{e}}$ in f_n.

Non-uniform VPSPACE is then defined by replacing PSPACE by its non-uniform analogue, PSPACE/ poly. Since there are no monotone analogues of Turing machines, perhaps the only possible monotone analogue of this definition is to insist on the coefficient function being monotone, which results in an absurdly weak class (the "largest" monomial will always be present).

Mahajan and Rao [MR13] look at the notion of *width* of a circuit — all gates are assigned heights, such that the height of any gate is *exactly* one larger than the height of its highest child. The width of the circuit is the maximum number of nodes that have the same height. They then define $\text{VSPACE}(S(n))$, as the class of families that are computable by circuits of width $S(n)$ and size at most $\max\left\{2^{S(n)}, \text{poly}(n)\right\}$.

The class uniform $\text{VSPACE}(S(n))$ further requires that the circuits be $\text{DSPACE}(\mathsf{S}(\mathsf{n}))$-uniform. Although their non-uniform definition is purely algebraic, it is a bit unnatural for space $S(n) \gg \log n$ (as also pointed out in their paper), since such circuits may not even have a $\text{poly}(n)$-sized description. We therefore do not analyse a monotone analogue for their definition.