




Monotone Classes Beyond VNP

Prerona Chatterjee   

Blavatnik School of Computer Science, Tel Aviv University, Israel

Kshitij Gajjar  

Indian Institute of Technology Jodhpur, Rajasthan, India

Anamay Tengse   

Department of Computer Science, University of Haifa, Israel

Abstract

In this work, we study the natural monotone analogues of various equivalent definitions of VPSPACE: a well studied class (Poizat 2008, Koiran & Perifel 2009, Malod 2011, Mahajan & Rao 2013) that is believed to be larger than VNP. We observe that these monotone analogues are not equivalent unlike their non-monotone counterparts, and propose *monotone* VPSPACE (mVPSPACE) to be defined as the monotone analogue of Poizat’s definition. With this definition, mVPSPACE turns out to be exponentially stronger than mVNP and also satisfies several desirable closure properties that the other analogues may not.

Our initial goal was to understand the monotone complexity of *transparent polynomials*, a concept that was recently introduced by Hrubeš & Yehudayoff (2021). In that context, we show that transparent polynomials of large sparsity are hard for the monotone analogues of all the known definitions of VPSPACE, except for the one due to Poizat.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Complexity, Monotone Computation, VPSPACE, Transparent Polynomials

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.11

Related Version *Full Version on Arxiv:* <https://arxiv.org/abs/2202.13103>

Funding *Prerona Chatterjee:* Azrieli International Postdoctoral Fellowship, the Israel Science Foundation (grant number 514/20) and the Len Blavatnik and the Blavatnik Family foundation. Parts of this work was done while I was a PhD student at TIFR Mumbai, where I was partially supported by a Google PhD Fellowship and also while I was a postdoctoral researcher at the Czech Academy of Sciences, where I was supported by Grant GX19-27871X of the Czech Science Foundation.

Kshitij Gajjar: Part of this work was done as a postdoc in NUS, where I was funded by NUS ODPRT Grant WBS No. R-252-000-A94-133.

Anamay Tengse: Grant 716/20 of the Israel Science Foundation.

Acknowledgements We would like to thank the anonymous reviewers for the helpful comments. We are also grateful to Meena Mahajan for suggestions that greatly improved the presentation of the paper.

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. A formal definition can be



© Prerona Chatterjee, Kshitij Gajjar, and Anamay Tengse;
licensed under Creative Commons License CC-BY 4.0

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).

Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 11; pp. 11:1–11:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

found in Section 2. The central question in the area is to show super-polynomial lower bounds against algebraic circuits for *explicit* polynomials, or equivalently, to show that $\text{VP} \neq \text{VNP}$: 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 [21, 22, 23, 12, 13, 7]. The most well-known among these, is the result of Jerrum & Snir [12], 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 [20, 6, 1].

Additionally, many separations that are believed to be true in the general setting have actually been proved to be true in the monotone setting [22, 9, 26, 24]. Most remarkably, Yehudayoff [26] showed an exponential separation between the computational powers of the monotone analogues of VP and VNP. We denote these classes by mVP (Definition 2.3) and mVNP (Definition 2.4) respectively.

Another line of work in this setting tries to understand the power of non-monotone computational models while computing monotone polynomials. Valiant [25], 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 [10, 4, 3, 5] 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 $\text{Newt}(f)$, is the convex hull in \mathbb{R}^n of the *exponent vectors* of the monomials in the support of f . Recently, Hrubeš & Yehudayoff [11] proposed studying the *Shadows of Newton polytopes* (projections to two-dimensional planes) as an approach to refute the τ -conjecture for Newton polygons made by Koiran, Portier, Tavenas & Thomassé [16].

Informally, the τ -conjecture for Newton polygons [16] 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 $\text{poly}(s, r, p)$ vertices. A formal definition of Newton polytopes and the τ -conjecture for Newton polygons can be found in Appendix A.

This conjecture is fairly strong, and it implies, among other things, that $\text{VP} \neq \text{VNP}$. 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 & Yehudayoff [11] 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* (maximum number of vertices in its projection), 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 & 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 [11] were not able to actually use this approach to refute the conjecture, they used the notions of shadows & 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** ([11, Theorem 2]). *If f is transparent then every monotone circuit computing f has size at least $\Omega(|\text{supp}(f)|)$.*

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. Throughout this work we assume that the underlying field is either the field of real numbers or the field of rational numbers. The goal of this work is two-fold.

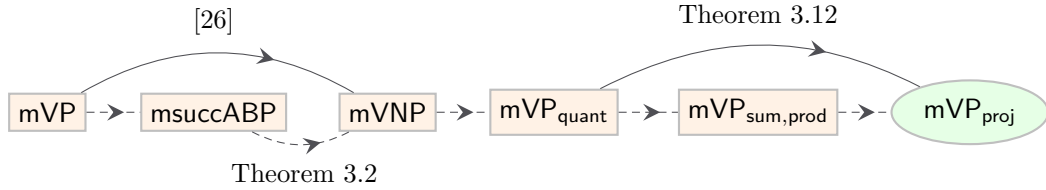
The first goal is to understand how restrictive the notion of transparency is. Our search begins with an observation by Yehudayoff [26], that any lower bound against mVP depending solely on the support of the hard polynomial, automatically “lifts” to mVNP with the same parameters¹. Since transparency 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 suspect that transparency is an even stronger property. Therefore, a natural question for us 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. We therefore turn to the literature in the non-monotone setting. Here, VPSPACE is a well studied class [19, 14, 18, 17]

¹ [26]: “If a monotone circuit-size lower bound for $q(\mathbf{x})$ holds also for all polynomials that are equivalent to $q(\mathbf{x})$ then the same lower bound also holds for every mVNP circuit computing $q(\mathbf{x})$.” Here mVNP circuit denotes $\sum_{\mathbf{z} \in \{0,1\}^m} \mathcal{C}(\mathbf{x}, z_1, \dots, z_m)$ where $m = \text{poly}(n)$ and $\mathcal{C}(\mathbf{x}, \mathbf{z})$ is a monotone algebraic circuit.

that is believed to be strictly larger than VNP. Interestingly there are multiple definitions of VPSPACE, resulting from varied motivations, all of which are known to be essentially equivalent [18, 17]. We study the natural monotone analogues of these definitions and show that unlike the non-monotone setting, the powers of the different resulting models vary greatly. This allows us to then analyse if the technique of Hrubeš & Yehudayoff also works against monotone classes that are possibly larger than mVNP.

The following figure succinctly describes some of our main results.



■ **Figure 1** Nodes represent classes of polynomial families; $A \dashrightarrow B \equiv A \subseteq B$ and $A \longrightarrow B \equiv A \subsetneq B$. Transparent polynomials are hard for all models corresponding to orange, rectangular nodes.

In Figure 1, the node labels refer to the following classes of polynomial families that have degree-poly(n) and poly(n)-complexity under the corresponding models.

- msuccABP - monotone succinct ABPs (Definition 3.1),
- mVP_{quant} - quantified monotone circuits (Definition 3.4),
- mVP_{sum,prod} - monotone circuits with summation and production gates (Definition 3.8),
- mVP_{proj} - monotone circuits with projection gates (Definition 3.11).

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 3.10).

An interesting point to note here is that there is an exponential separation between mVP_{quant} and mVP_{proj}, which means that at least one of the inclusions: mVP_{quant} to mVP_{sum,prod}, and mVP_{sum,prod} to mVP_{proj} is strict with an exponential separation.

Additionally, we show the following two statements about mVP_{quant}.

- mVP_{quant} = mVNP if and only if homogeneous components of polynomials in mVP_{quant} are contained in mVNP (Corollary 3.6). In particular, we show that homogeneous polynomials in mVP_{quant} are also in mVNP (Theorem 3.5).
- mVP_{quant} = mVP_{sum,prod} if and only if quantified monotone circuits are closed under compositions (Observation 3.9).

Finally, we also show that the homogeneous components of polynomials in mVP_{proj} are in mVP_{proj} (Theorem 3.13). This property, along with the fact that Perm _{n} ∈ mVP_{proj} (Theorem 6.1), is the reason we propose “monotone VPSPACE” (mVPSPACE) to be defined as the class of polynomial families that are efficiently computable by monotone circuits with projection gates (without any restriction on degree).

1.2 Organization 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 and Section 6. We conclude with Section 7, where we discuss some 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 $[n] = \{1, 2, \dots, n\}$.
- 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: $\mathbf{e} = (e_1, e_2, \dots, e_n)$, $\mathbf{x} = \{x_1, \dots, x_n\}$. 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 $x_1^{e_1} x_2^{e_2} \dots x_n^{e_n}$.
- For a polynomial $f(\mathbf{x})$, we denote by $\deg(f)$ the degree of f in \mathbf{x} .
- For a polynomial $f(\mathbf{x})$ and a monomial $m = \mathbf{x}^{\mathbf{e}}$, we refer to the coefficient of m in f by $\text{coeff}_f(m)$. The support $\text{supp}(f)$ of a polynomial f is given by $\{m : \text{coeff}_f(m) \neq 0\}$, and the *sparsity* of a polynomial is the size of its support, $|\text{supp}(f)|$.
- For any polynomial $f(\mathbf{x})$ and any $k \leq \deg(f)$, we denote by $\text{hom}_k(f)$ the k -th homogeneous degree component of f in terms of \mathbf{x} . That is, if $f(\mathbf{x}) = f_0(\mathbf{x}) + \dots + f_{\deg(f)}(\mathbf{x})$ where $f_k(\mathbf{x})$ is a homogeneous polynomial of degree k in \mathbf{x} , then $\text{hom}_k(f) = f_k$.
- The permanent of an $n \times n$ symbolic matrix shall be denoted by Perm_n and is defined as $\text{Perm}_n = \sum_{\sigma \in S_n} \prod_{i=1}^n x_{i, \sigma(i)}$, where S_n is the set of all permutations of $[n]$.
- We use $\{f_n\}$ to denote families of polynomials indexed by \mathbb{N} . All complexity classes are defined in terms of asymptotic properties of “polynomials” and are technically sets of such polynomial families. Sometimes however, this technicality is ignored for the sake of brevity, especially when the analogous statement about polynomial families is obvious.

► **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) have in-degree 2.*

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

In case there are multiple output gates, the circuit is said to be multi-output, and computes a set of polynomials.

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.

► **Definition 2.2** (Algebraic Branching Programs (ABPs)). *An algebraic branching program is specified by a layered graph where each edge is labelled by an affine linear form and the first and the last layer have one vertex each, called the “source” and the “sink” vertex respectively. The polynomial computed by an ABP is equal to the sum of the weights of all paths from the start vertex to the end vertex in the ABP, where the weight of a path is equal to the product of the labels of all the edges on it.*

The width of a layer in an ABP is the number of vertices in it and the width of an ABP is the width of the layer that has the maximum number of vertices in it.

The size of an ABP is the number of vertices in it.

Basic monotone classes

► **Definition 2.3** (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 2.4** (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 an m -variate family $\{g_m\} \in \text{mVP}$ with $m, \text{size}(g_m) \leq n^c$, such that for all large enough n , f_n satisfies the following.

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

An expression of the above form is alternatively called an *exponential sum* computing f_n .

Various definitions of VPSPACE

Koiran & Perifel [14, 15] 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 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 $\text{VP} \neq \text{VPSPACE}_b$ then either $\text{VP} \neq \text{VNP}$ or $\text{P/poly} \neq \text{PSPACE/poly}$.

Later, Poizat [19] 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 2.5** (Projection gates [19]). A projection gate is a unary gate that is labelled by a variable z and a constant $b \in \{0, 1\}$, denoted by $\text{fix}_{(z=b)}$. It returns the partial evaluation of its input polynomial, at $z = b$, that is, $\text{fix}_{(z=b)}(f(z, \mathbf{x})) = f(b, \mathbf{x})$.

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 showed² that this definition is equivalent to that of Koiran & Perifel.

► **Definition 2.6** (Algebraic circuits with projection gates [19]). An algebraic circuit with projection gates is an algebraic circuit (Definition 2.1) in which the internal nodes can also be projection gates (Definition 2.5), in addition to $+$ or \times .

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

Adding to Poizat's work, Malod [18] characterized 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 $\text{poly}(n)$.

► **Definition 2.7** (Succinct ABPs [18]). A succinct ABP over the n variables $\mathbf{x} = \{x_1, \dots, x_n\}$ is a triple $(B, \mathbf{s}, \mathbf{t})$ 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.

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 succinct ABP.

² The work of Poizat is written in French, Malod [18] provides an alternate exposition of some of the main results in English.

In the same work [18], Malod alternatively characterized VPSPACE using an interesting algebraic model that resembles *(totally) quantified boolean formulas* that are known to characterize 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 2.8** (Summation and Production gates [18]). *Summation and production gates are unary gates that are labelled by a variable z , and are denoted by sum_z and 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, $\text{sum}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) + f(1, \mathbf{x})$, and $\text{prod}_z(f(z, \mathbf{x})) = f(0, \mathbf{x}) \cdot f(1, \mathbf{x})$.*

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

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

► **Definition 2.9** (Quantified Algebraic Circuits [18]). *A quantified algebraic circuit is an algebraic circuit that has the form,*

$$Q_{z_1}^{(1)} Q_{z_2}^{(2)} \cdots Q_{z_m}^{(m)} C(\mathbf{x}, \mathbf{z}),$$

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

Finally, Mahajan & Rao [17] 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 relationships known in the boolean world to the algebraic world (e.g. they show $\text{VL} \subseteq \text{VP}$). They further show that their definition of uniform polynomially-bounded-space computation coincides with that of uniform-VPSPACE as defined by Koiran & Perifel [14].

We now narrow our focus to the definitions due to Poizat [19] and Malod [18]. 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 B.

► **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 [14], unless mentioned otherwise.

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.

3.1 Monotone succinct ABPs

We first consider the natural monotone analogue of the definition due to Malod [18] which uses succinct algebraic branching programs (Definition 2.7).

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}^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 predefined³ 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 the n variables $\mathbf{x} = \{x_1, \dots, x_n\}$ 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 $f \in \text{mVP}$ 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 $\mathcal{C}'(u, v, \mathbf{x}) = v \cdot \mathcal{C}(\mathbf{x})$.

We show that the computational power of monotone succinct ABPs when computing polynomials of *bounded degree* does not even go beyond mVNP .

► **Theorem 3.2.** *If a polynomial family $\{f_n\}$ of degree $\text{poly}(n)$ is computable by monotone succinct ABPs of complexity $\text{poly}(n)$, then $\{f_n\} \in \text{mVNP}$.*

In contrast, Malod [18] showed that every family in VPSPACE admits succinct ABPs of polynomial complexity, and we expect VPSPACE_b to be a much bigger class than VNP . The proof of Theorem 3.2 is quite straightforward and relies on the following claim.

▷ **Claim 3.3.** For any f_n , let $\mathcal{A} = (B, \mathbf{s}, \mathbf{t}, \ell)$ be the monotone succinct ABP computing it, with $|\mathbf{s}| = |\mathbf{t}| = r$. If $\ell > 1$, then $\ell \leq \deg(f_n) + 2$.

This bound allows us to write the sum of all s to t paths in the ABP as an exponential sum of an mVP expression, finishing the proof. A complete proof can be found in Subsection C.4.

It is not clear to us if the converse of Theorem 3.2 is true. Any obvious attack seems to fail due to the restriction that the circuit encoding the ABP needs to be monotone.

3.2 Quantified monotone circuits

As mentioned earlier, Malod [18] had also characterized the class VPSPACE using the notion of quantified algebraic circuits (Definition 2.9). We now consider its natural monotone analogue, which we call quantified monotone circuits.

³ It is not hard to see that the analogous definition in the non-monotone setting is equivalent to Malod's definition (Definition 2.7). This is essentially because of the connection to Iterated Matrix Multiplication.

► **Definition 3.4** (Quantified Monotone Algebraic Circuits). *A quantified monotone algebraic circuit has the form*

$$Q_{z_1}^{(1)} Q_{z_2}^{(2)} \dots Q_{z_m}^{(m)} C(\mathbf{x}, \mathbf{z})$$

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

We denote by $\text{mVP}_{\text{quant}}$ the class of all n -variate polynomial families of degree $\text{poly}(n)$ that are computable by quantified monotone algebraic circuits of size $\text{poly}(n)$.

Clearly $\text{mVNP} \subseteq \text{mVP}_{\text{quant}}$. It is therefore interesting to check if the inclusion is tight. We show that $\text{mVNP} \neq \text{mVP}_{\text{quant}}$ if and only if there is a family $\{f_n\} \in \text{mVP}_{\text{quant}}$ such that the k -th homogeneous component of f_n is not in $\text{mVP}_{\text{quant}}$ for some n and $k \leq \deg(f)$.

In particular we show the following statement.

► **Theorem 3.5.** *Let f be computable by a quantified monotone circuit of size s . If f is homogeneous, then it is expressible as an exponential sum of size at most $O(s \cdot \deg(f))$.*

Since mVNP is closed under addition, we get the following as a corollary.

► **Corollary 3.6.** *The class $\text{mVP}_{\text{quant}}$ is closed under taking homogeneous components, if and only if, $\text{mVP}_{\text{quant}} = \text{mVNP}$. That is,*

$$(\forall f \in \text{mVP}_{\text{quant}}, \forall k \leq \deg(f), \text{hom}_k(f) \in \text{mVP}_{\text{quant}}) \iff \text{mVNP} = \text{mVP}_{\text{quant}}$$

A proof of Theorem 3.5 and Corollary 3.6 can be found in Section 4.

Even though we believe $\text{mVNP} \subsetneq \text{mVP}_{\text{quant}}$, we feel this might be tricky to prove. The following theorem sheds some light on why that may be the case.

► **Theorem 3.7.** *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 monotone polynomial $A(\mathbf{w})$ such that,*

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

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

Although the obvious size and degree bounds on $A(\mathbf{w})$ above are exponential, it has a somewhat succinct quantified expression that can be inferred from the proof (see Subsection C.5). We now discuss how Theorem 3.7 helps us understand a possible difficulty in separating $\text{mVP}_{\text{quant}}$ from mVNP .

1. If the polynomial $A(\mathbf{w})$ from Theorem 3.7 were to have degree and size that is polynomial in n , then $\text{mVP}_{\text{quant}}$ 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 $f(\mathbf{x}) = \sum_{\mathbf{b}} B(\mathbf{b})h(\mathbf{x}, \mathbf{b})$. 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 $A(\mathbf{w})$ which is hard to compute for mVNP , even as a function over the boolean hypercube; a result that perhaps stands on its own.

11:10 Monotone Classes Beyond VNP

A proof sketch of Theorem 3.7 can be found in Section 4 and a complete proof can be found in Subsection C.5.

3.3 Monotone circuits with summation and production gates

Note that it is unclear if quantified monotone circuits are closed under compositions. We therefore also consider a model that generalizes quantified monotone circuits and is trivially closed under compositions. Here summation and production gates are allowed to appear anywhere in the circuit.

► **Definition 3.8** (Algebraic circuits with summation and production gates). *An algebraic circuit with summation and production gates is an algebraic circuit (Definition 2.1) in which the internal nodes can also be summation or production gates (Definition 2.8), in addition to $+$ or \times . A subset of the variables used by the circuit are marked as auxiliary. These variables do not appear in the output polynomial(s) of the circuit, and the labels for all the summation and production gates are required to be auxiliary variables.*

The size of an algebraic circuit with summation and production gates 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.

We denote by $\text{mVP}_{\text{sum,prod}}$ the class of all n -variate polynomial families of degree $\text{poly}(n)$ that are computable by monotone algebraic circuits with summation and production gates of size $\text{poly}(n)$.

Note that even in the non-monotone setting this model is clearly as powerful as quantified circuits, but can be simulated by circuits with projection gates. Again, Malod [18] showed that quantified circuits and circuits with projection gates are equivalent in power. So the class of polynomials efficiently computable by this model is also 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, we observe the following. Here, we mean “closure under compositions” in a strong sense: if C_1 and C_2 are quantified monotone circuits of size s_1 and s_2 respectively, then the polynomial computed by their composition to have a quantified monotone circuit of size at most $s_1 + s_2$.

► **Observation 3.9** (Informal). *Quantified monotone circuits are closed under compositions, if and only if, $\text{mVP}_{\text{quant}} = \text{mVP}_{\text{sum,prod}}$.*

Theorem 5.1 gives a formal statement and its proof can be found in Subsection C.3.

We, however, show that even this seemingly stronger model does not help in computing transparent polynomials.

► **Theorem 3.10.** *Any monotone algebraic circuit with summation and production gates that computes a transparent polynomial f , has size at least $|\text{supp}(f)|/4$.*

This shows that transparent polynomials with large support are hard even for this model. A proof sketch can be found in Section 5.

Recall that one way to refute the τ -conjecture for Newton polygons is to show a transparent polynomial in (non-monotone) VP . Theorem 3.10 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 [10, 4], such a result would also be quite interesting and would further highlight the power of subtractions.

3.4 Monotone circuits with projection gates

Finally, adapting the definition of VPSPACE due to Poizat (Definition 2.6) [19], we define monotone circuits with projection gates.

► **Definition 3.11** (Monotone algebraic circuits with projection gates). *A monotone algebraic circuit with projection gates is an algebraic circuit with projections (Definition 2.6) in which only non-negative constants from the field are allowed to appear as labels of leaves.*

As in Definition 3.8, only the auxiliary variables can be used as labels for the projection gates. The size of a monotone algebraic circuit with projection gates is the number of nodes in the underlying graph.

We denote by mVP_{proj} the class of all n -variate polynomials of degree $\text{poly}(n)$ that are computable by size- $\text{poly}(n)$ monotone algebraic circuits with projection gates.

This model is clearly at least as powerful as monotone circuits with summation and production gates, since $\text{sum}_z = \text{fix}_{(z=0)} + \text{fix}_{(z=1)}$ and $\text{prod}_z = \text{fix}_{(z=0)} \times \text{fix}_{(z=1)}$. 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 $2^{\Omega(\sqrt{m})}$ separation.

► **Theorem 3.12.** *The polynomial family $\{\text{Perm}_n\}$ can be computed by monotone circuits with projection gates of size $O(n^3)$, but quantified monotone circuits computing it must have size $2^{\Omega(n)}$.*

Finally we show that mVP_{proj} is closed under taking homogeneous components.

► **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$, $\text{hom}_k(f)$ has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

Proof sketches of Theorem 3.12 and Theorem 3.13 can be found in Section 6.

3.5 Defining Monotone VPSPACE (mVPSPACE)

We propose the following definition for mVPSPACE .

► **Definition 3.14** (Monotone VPSPACE). *A family of polynomials $\{f_n\}$ is said to be in mVPSPACE if for all large n , f_n is computable by a monotone algebraic circuit with projection gates (Definition 3.11) of size $\text{poly}(n)$.*

Further if $\{f_n\}$ has degree $\text{poly}(n)$, then it is said to be in mVPSPACE_b .

That is, we define $\text{mVPSPACE}_b := \text{mVP}_{\text{proj}}$ and define mVPSPACE along the same lines, but without the restriction on the degree being bounded (since VPSPACE does not impose any restrictions on degree). Some of our reasons for this choice are as follows.

Firstly, being a complexity class, mVPSPACE_b should be closed under (monotone) affine projections, i.e. setting a few variables to monotone affine polynomials. All of $\text{mVP}_{\text{quant}}$, $\text{mVP}_{\text{sum,prod}}$ and mVP_{proj} have this property.

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

11:12 Monotone Classes Beyond VNP

Further, as mVP and mVNP are closed under taking homogeneous components, it is desirable for a more powerful class to also have this property. Even if $\text{mVP}_{\text{quant}}$ satisfies this, it would not lead to a larger class (Corollary 3.6). Also, it is not clear $\text{mVP}_{\text{sum,prod}}$ is closed under homogenization, while mVP_{proj} is (Theorem 3.13).

Finally, we believe that having $\text{Perm}_n \in \text{mVP}_{\text{proj}}$ is an interesting property that further strengthens the case for mVP_{proj} being the definition for mVPSPACE_b .

4 Quantified monotone circuits

Computing homogeneous polynomials

► **Theorem 3.5.** *Let f be computable by a quantified monotone circuit of size s . If f is homogeneous, then it is expressible as an exponential sum of size at most $O(s \cdot \deg(f))$.*

Proof. Let $d = \deg(f)$, and let \mathcal{C} be a quantified monotone circuit computing f , that uses exactly k production gates. We can then assume that,

$$\mathcal{C}(\mathbf{x}) = \sum_{\mathbf{y}_0} \prod_{z_1} \sum_{\mathbf{y}_1} \prod_{z_2} \cdots \sum_{\mathbf{y}_{k-1}} \prod_{z_k} \sum_{\mathbf{y}_k} g(\mathbf{x}, \mathbf{y}, \mathbf{z}),$$

without loss of generality, by using some empty \mathbf{y}_j s whenever necessary. Note that the \mathbf{y}_j s are sets of variables, whereas each of the z_j s are single variables.

We now prove the statement in two steps. First, we use the homogeneity of f , and the monotonicity of the quantified circuit, to show that $k \leq \log(d)$.

▷ **Claim 4.1.** $k \leq \log d$

Proof. For each $i \in [k]$, let $g_i(z_i, \mathbf{x}, \mathbf{w}_i) = \sum_{\mathbf{y}_i} \prod_{z_{i+1}} \sum_{\mathbf{y}_{i+1}} \cdots \sum_{\mathbf{y}_k} g(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Here \mathbf{w}_i denotes all the auxiliary variables that are alive after “ i rounds” of quantifiers. Further, let $h_i(\mathbf{x}, \mathbf{w}_i) = \prod_{z_i} g_i(z_i, \mathbf{x}, \mathbf{w}_i)$.

Now, $f(\mathbf{x}) = \sum_{\mathbf{y}_0} h_1(\mathbf{x}, \mathbf{y}_0)$, and it is homogeneous. Therefore, since h_1 is monotone, it is also homogeneous in \mathbf{x} with degree exactly d . But $\deg_{\mathbf{x}}(h_1) = \deg_{\mathbf{x}}(\prod_{z_1} g_1) = \deg_{\mathbf{x}}(g_1(z_1 = 0)) + \deg_{\mathbf{x}}(g_1(z_1 = 1))$. If we write $g_1(z_1, \mathbf{x}, \mathbf{w}_1) = g_{1,0}(\mathbf{x}, \mathbf{w}_1) + z_1 \cdot g_{1,1}(z_1, \mathbf{x}, \mathbf{w}_1)$, then we have that $g_1(z_1 = 0) = g_{1,0}(\mathbf{x}, \mathbf{w}_1)$ and $g_1(z_1 = 1) = g_{1,0}(\mathbf{x}, \mathbf{w}_1) + g_{1,1}(z_1 = 1, \mathbf{x}, \mathbf{w}_1)$. Since h_1 is homogeneous in \mathbf{x} and g_1 is monotone in all the variables, this must mean that $\deg_{\mathbf{x}}(g_1(z_1 = 0)) = \deg_{\mathbf{x}}(g_1(z_1 = 1)) = \deg_{\mathbf{x}}(g_1) = d/2$. Also, g_1 is homogeneous in \mathbf{x} , and thus we can repeat the same argument for h_2, g_2 , and so on.

As a result, we see that $\deg(f) = 2^k \cdot \deg_{\mathbf{x}}(g)$, and hence $k \leq \log d$. ◁

We can now make $2^k \leq d$ many copies of the “inner circuit” $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$, one for each fixing of the \mathbf{z} variables. We then obtain the final exponential sum computing f by using the following “product rule” for summations repeatedly.

$$(\sum_{\mathbf{y}_1} h_1(\mathbf{x}, \mathbf{y}_1)) \cdot (\sum_{\mathbf{y}_2} h_2(\mathbf{x}, \mathbf{y}_2)) = \sum_{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2} (h_1(\mathbf{x}, \tilde{\mathbf{y}}_1) \cdot h_2(\mathbf{x}, \tilde{\mathbf{y}}_2))$$

Note that in the above case the two summations are over disjoint sets of variables. This can easily be ensured in our case, by treating the \mathbf{y} variables in each of the $2^k \leq d$ copies as mutually disjoint. It is easy to see that the exponential sum has size $O(\text{size}(C), d)$. ◀

► **Remark 4.2.** The first step in the above proof extends more or less as it is, to an arbitrary circuit with summation and production gates. Thus, any circuit with arbitrary summations and productions *that computes a homogeneous polynomial* can be assumed to not contain any production gates, with a polynomial blow-up in size.

However, this does not directly give an efficient exponential sum, because of the second step in the above argument. It crucially uses the fact that for any summation gate g , the number of production gates on a path from g to the root was $O(\log d)$. This ensures that no summation gate (or its auxiliary variable) has to be replicated more than $\text{poly}(d)$ times, which is not necessarily true if we start with an arbitrary circuit with summation gates.

Large exponential sums for arbitrary polynomials

► **Theorem 3.7.** *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 monotone polynomial $A(\mathbf{w})$ such that,*

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

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

We shall need the following simple observation, which follows from the “product-rule” for summations stated earlier.

► **Observation 4.3** (Product of exponential sums).

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

Let us see a toy case of trivially moving from a quantified expression to an exponential sum, using Observation 4.3.

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

In the last line, each $*$ 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. 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.

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

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

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 that allows us to prove Theorem 3.7 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 $\mathbf{a} \in \{0, 1\}^r$. A complete proof of Theorem 3.7 can be found in Subsection C.5.

5 Monotone circuits with summation and production gates

In this section, we give the proof overview of Theorem 3.10.

► **Theorem 3.10.** *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 & Yehudayoff [11]. 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 (in terms of the number of 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 [11] is that if the sum or product of two monotone polynomials is convexly independent, then so are each of the two inputs. However, 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 sum_z produces just $\{y_1y_2\}$, which is convexly independent.

In order to prove Theorem 3.10, one can get around this by working directly with the support projected down to the “true” variables, which we call \mathbf{y} -support in our arguments. It turns out that summations and productions indeed preserve convex dependencies that are in the \mathbf{y} support of the input polynomial. Since the proof follows exactly along the same lines as the one in [11], we omit the proof here. A formal proof can be found in the full version [2].

Quantified monotone circuits and compositions

► **Observation 3.9 (Informal).** *Quantified monotone circuits are closed under compositions, if and only if, $\text{mVP}_{\text{quant}} = \text{mVP}_{\text{sum,prod}}$.*

Even though this statement appears to be straightforward, formally stating it requires a bit more care. In particular, we require quantified circuits to be closed under compositions in a strong sense, similar to usual algebraic circuits. Doing that yields the following theorem, which we prove in Subsection C.3.

► **Theorem 5.1.** *Suppose that for any multi-output quantified monotone circuit \mathcal{C} of size s with r inputs, and any multi-output quantified monotone circuit \mathcal{C}' of size s' with r outputs, we have that the polynomial computed by $\mathcal{C} \circ \mathcal{C}'$ has a quantified monotone circuit of size at most $(s + s')$.*

Then, any multi-output, monotone circuit with summation and production gates of size \tilde{s} can be simulated by a multi-output quantified monotone circuit of size at most \tilde{s} , and hence $\text{mVP}_{\text{quant}} = \text{mVP}_{\text{sum,prod}}$.

The converse is also true.

6 Monotone circuits with projection gates

Exponential separation from quantified circuits

► **Theorem 3.12.** *The polynomial family $\{\text{Perm}_n\}$ can be computed by monotone circuits with projection gates of size $O(n^3)$, but quantified monotone circuits computing it must have size $2^{\Omega(n)}$.*

We begin by proving that $\text{Perm}_n \in \text{mVP}_{\text{proj}}$.

► **Theorem 6.1.** *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 exactly 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 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 j th column by projecting the n variables $y_{1,j}, \dots, y_{n,j}$.

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

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

The following claim is now easy to verify.

► **Claim 6.2.** 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 j th 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 $j+1, \dots, n$; thus $P_n = \text{Perm}_n$.

The size of our circuit is $O(n^3)$, since $\text{size}(P_0) = O(n^2)$ and $\text{size}(P_j) = \text{size}(P_{j-1}) + O(n^2)$. This proves Theorem 6.1. ◀

► **Remark 6.3.** Our upper bound above also implies that any polynomial (family) that can be expressed as the permanent of a monotone matrix of size $\text{poly}(n)$ (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 [8].

The proof of Theorem 3.12 now follows from the following simple extension of an observation due to Yehudayoff [26] and the classical lower bound of Jerrum & Snir [12] against monotone algebraic circuits for Perm_n . A complete proof can be found in Subsection C.1.

► **Lemma 6.4.** *Let $f(\mathbf{x})$ be a monotone polynomial whose support cannot be written as a non-trivial product of two sets, and for some monotone polynomial $g(\mathbf{x}, \mathbf{z})$, suppose we have $f(\mathbf{x}) = \mathbb{Q}_{z_1}^{(1)} \mathbb{Q}_{z_2}^{(2)} \cdots \mathbb{Q}_{z_m}^{(m)} g(\mathbf{x}, \mathbf{z})$ with $\mathbb{Q}^{(i)} \in \{\text{sum}, \text{prod}\}$ for each $i \in [m]$.*

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

Closure under homogenization

► **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$, $\text{hom}_k(f)$ has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

We show this using the classical argument of “gate replication” and the complete proof can be found in Subsection C.2.

7 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 & Yehudayoff [11] 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. The following are some interesting open threads from our work.

- We suspect 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 size- s monotone circuit with projection gates, then $|\text{supp}(f)| \leq 2^{\text{polylog}(s)}$. It would be interesting (at least to us) to see a proof or a refutation of this conjecture. An immediate hurdle in extending the techniques in [11] (as in Theorem 3.10) to mVPSPACE, is that unlike summations and productions, 0-projections do not preserve convex dependencies, even if we restrict to the “true” variables.
- Along similar lines, a possibly simpler goal is to show a non-monotone circuit upper bound for a transparent polynomial. Since transparency only restricts the support of the polynomial, one is free to choose any real coefficients to ensure that it is in VP. In particular, this brings powerful non-monotone tricks like interpolation into play. 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 3.7, 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

- 1 Bruno Pasqualotto Cavalari, Mrinal Kumar, and Benjamin Rossman. Monotone circuit lower bounds from robust sunflowers. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *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. doi:10.1007/978-3-030-61792-9_25.
- 2 Prerona Chatterjee, Kshitij Gajjar, and Anamay Tengse. Monotone classes beyond VNP. *CoRR*, abs/2202.13103, 2022. arXiv:2202.13103.
- 3 Arkadev Chattopadhyay, Rajit Datta, Utsab Ghosal, and Partha Mukhopadhyay. Monotone complexity of spanning tree polynomial re-visited. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPICs*, pages 39:1–39:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.39.

- 4 Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Lower bounds for monotone arithmetic circuits via communication complexity. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 786–799. ACM, 2021. doi:10.1145/3406325.3451069.
- 5 Arkadev Chattopadhyay, Utsab Ghosal, and Partha Mukhopadhyay. Robustly separating the arithmetic monotone hierarchy via graph inner-product. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPICs*, pages 12:1–12:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.FSTTCS.2022.12.
- 6 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.
- 7 S.B. Gashkov. The complexity of monotone computations of polynomials. *Moscow University Math Bulletin*, 42(5):1–8, 1987.
- 8 Joshua A. Grochow. Monotone projection lower bounds from extended formulation lower bounds. *Theory of Computing*, 13(18):1–15, 2017. doi:10.4086/toc.2017.v013a018.
- 9 Pavel Hrubeš and Amir Yehudayoff. On isoperimetric profiles and computational complexity. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *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. doi:10.4230/LIPICs.ICALP.2016.89.
- 10 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. doi:10.1109/CCC.2013.11.
- 11 Pavel Hrubeš and Amir Yehudayoff. Shadows of newton polytopes. In Valentine Kabanets, editor, *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. doi:10.4230/LIPICs.CCC.2021.9.
- 12 Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *Journal of the ACM*, 29(3):874–897, 1982. doi:10.1145/322326.322341.
- 13 O. M. Kasim-Zade. Arithmetic complexity of monotone polynomials. *Theoretical Problems in Cybernetics. Abstracts of lectures*, pages 68–69, 1986.
- 14 Pascal Koiran and Sylvain Perifel. VPSPACE and a transfer theorem over the reals. *Computational Complexity*, 18(4):551–575, 2009. doi:10.1007/s00037-009-0269-1.
- 15 Pascal Koiran and Sylvain Perifel. Vpspace and a transfer theorem over the complex field. *Theoretical Computer Science*, 410(50):5244–5251, 2009. Mathematical Foundations of Computer Science (MFCS 2007). doi:10.1016/j.tcs.2009.08.026.
- 16 Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A τ -conjecture for newton polygons. *Foundations of Computational Mathematics*, 15:185–197, 2015. doi:10.1007/s10208-014-9216-x.
- 17 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. doi:10.1007/s00037-011-0024-2.
- 18 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. doi:10.1007/978-3-642-22953-4_18.

- 19 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. doi:10.2178/jsl/1230396913.
- 20 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. doi:10.1016/j.jcss.2010.06.013.
- 21 Claus-Peter Schnorr. A lower bound on the number of additions in monotone computations. *Theor. Comput. Sci.*, 2(3):305–315, 1976. doi:10.1016/0304-3975(76)90083-9.
- 22 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.
- 23 Eli Shamir and Marc Snir. On the depth complexity of formulas. *Math. Syst. Theory*, 13:301–322, 1980. doi:10.1007/BF01744302.
- 24 Srikanth Srinivasan. Strongly exponential separation between monotone VP and monotone VNP. *ACM Transactions on Computing Theory*, 12(4):23:1–23:12, 2020. doi:10.1145/3417758.
- 25 Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980. doi:10.1016/0304-3975(80)90060-2.
- 26 Amir Yehudayoff. Separating monotone VP and VNP. In Moses Charikar and Edith Cohen, editors, *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. doi:10.1145/3313276.3316311.

A

 Deferred Formal Definitions and Statements

► **Definition A.1** (Newton polytopes). For a polynomial $f(\mathbf{x})$, its Newton polytope $\text{Newt}(f) \subseteq \mathbb{R}^n$, is defined as the convex hull of the exponent vectors of the monomials in its support.

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

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

► **Conjecture A.2** (τ conjecture for Newton polytopes [16]). 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.

B

 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 & Perifel [15, 14], and Mahajan & Rao [17].

Koiran & Perifel define uniform VPSPACE as the class of families $\{f_n\}$ of 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 & Rao [17] 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 $\max\{2^{S(n)}, \text{poly}(n)\}$.

The class uniform $\text{VSPACE}(S(n))$ further requires that the circuits be $\text{DSPACE}(S(n))$ -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.

C Deferred Proofs

C.1 Proof of Theorem 3.12

We first consider Lemma 6.4, which is a simple extension of the following observation due to Yehudayoff [26].

► **Observation C.1** ([26]). *Let $g(\mathbf{x}, z)$ be a monotone polynomial and let $c > 0$. Then for any monomial $\mathbf{x}^e z^j$ in the support of g , $\mathbf{x}^e \in \text{supp}(g(\mathbf{x}, z = c))$.*

We now formally prove Lemma 6.4. For sets of monomials A and B , their product is defined as $A \times B = \{m \cdot m' : m \in A, m' \in B\}$; a non-trivial product is when neither A nor B is just $\{1\}$.

► **Lemma 6.4.** *Let $f(\mathbf{x})$ be a monotone polynomial whose support cannot be written as a non-trivial product of two sets, and for some monotone polynomial $g(\mathbf{x}, \mathbf{z})$, suppose we have $f(\mathbf{x}) = \mathbb{Q}_{z_1}^{(1)} \mathbb{Q}_{z_2}^{(2)} \cdots \mathbb{Q}_{z_m}^{(m)} g(\mathbf{x}, \mathbf{z})$ with $\mathbb{Q}^{(i)} \in \{\text{sum}, \text{prod}\}$ for each $i \in [m]$.*

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

Proof. By Observation C.1, it is enough to show the statement of the lemma for $m = 1$. Therefore, suppose $f(\mathbf{x}) = \text{sum}_z g(\mathbf{x}, z)$. Then $f(\mathbf{x}) = g(\mathbf{x}, 0) + g(\mathbf{x}, 1)$, and hence $\text{supp}(f) = \text{supp}(g(\mathbf{x}, 1))$, 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 $\text{supp}(f)$ 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 $\text{supp}(f(\mathbf{x})) = \text{supp}(g(\mathbf{x}, 1))$ as claimed. ◀

Finally, let us complete the proof of Theorem 3.12.

► **Theorem 3.12.** *The polynomial family $\{\text{Perm}_n\}$ can be computed by monotone circuits with projection gates of size $O(n^3)$, but quantified monotone circuits computing it must have size $2^{\Omega(n)}$.*

Proof. Let us assume that there is a quantified monotone circuit of size s computing Perm_n . Then,

$$\text{Perm}_n(\mathbf{x}) = \mathbb{Q}_{z_1}^{(1)} \mathbb{Q}_{z_2}^{(2)} \cdots \mathbb{Q}_{z_m}^{(m)} f(\mathbf{x}, \mathbf{z})$$

for some $m \leq s$ and $\mathbb{Q}^{(i)} \in \{\text{sum}, \text{prod}\}$ for each $i \in [m]$.

Note that, by definition, $f(\mathbf{x}, \mathbf{z})$ is computable by a monotone algebraic circuit of size at most s and therefore $f(\mathbf{x}, \bar{1})$ is computable by a monotone algebraic circuit of size at most s . On the other hand, by Lemma 6.4, the support of $f(\mathbf{x}, \bar{1})$ is the same as that of Perm_n since Perm_n is irreducible. The required lower bound now follows from the fact that the $2^{\Omega(n)}$ lower bound proved by Jerrum & Snir [12] for Perm_n against monotone algebraic circuits, works for any polynomial that has support equal to the support of Perm_n . ◀

C.2 Proof of Theorem 3.13

► **Theorem 3.13.** *Suppose f is computed by a size s monotone circuit with projections. Then for any $k \leq \deg(f)$, $\text{hom}_k(f)$ has a monotone circuit with projections of size $O(k^2 \cdot s)$.*

Proof. We show this using the classical argument of “gate replication”. Given a circuit \mathcal{C} , we construct another circuit \mathcal{C}' that has $(k + 1)$ copies of each gate in \mathcal{C} . For a gate $g \in \mathcal{C}$, the corresponding gates g_0, g_1, \dots, g_k shall compute $\text{hom}_i([g])$ for each $i \leq k$, where $[g]$ is the polynomial computed at g . Here and throughout the proof, the degree of a polynomial always refers to its degree in the \mathbf{x} -variables.

The following can now be easily checked, using the fact that $[g]$ is always a monotone polynomial.

- If $[g]$ is a leaf labelled with a “true” variable x_i , then $[g_1] = x_i$ and $[g_i] = 0$ for all other i .
- If $[g]$ is any other leaf, then $[g_0] = [g]$ and $[g_i] = 0$ for all other i .
- If $[g] = [u] + [v]$, then $[g_i] = [u_i] + [v_i]$ for all i .
- If $[g] = \text{fix}_{(z=b)}[u]$, then $[g_i] = \text{hom}_i([g]) = \text{fix}_{(z=b)} \text{hom}_i([u]) = \text{fix}_{(z=b)}[u_i]$.
- If $[g] = [u] \times [v]$, then $[g_i] = \sum_{j \leq i} [u_j] \times [v_{i-j}]$, for each i .

The last case incurs the largest blow-up in size, which adds $O(k^2)$ many gates in \mathcal{C}' for one gate in \mathcal{C} . This finishes the proof. ◀

C.3 Proof of Theorem 5.1

► **Theorem 5.1.** *Suppose that for any multi-output quantified monotone circuit \mathcal{C} of size s with r inputs, and any multi-output quantified monotone circuit \mathcal{C}' of size s' with r outputs, we have that the polynomial computed by $\mathcal{C} \circ \mathcal{C}'$ has a quantified monotone circuit of size at most $(s + s')$.*

Then, any multi-output, monotone circuit with summation and production gates of size \tilde{s} can be simulated by a multi-output quantified monotone circuit of size at most \tilde{s} , and hence $\text{mVP}_{\text{quant}} = \text{mVP}_{\text{sum,prod}}$.

The converse is also true.

Proof. One direction of the implication is clearly true because circuits with (arbitrary) summation and production gates have the stated property by definition.

For the converse, let us assume that quantified monotone circuits have the property. We show that this implies that the two models in question have the same power.

Consider a circuit \mathcal{C} of size s with summation and production gates. We group the gates in \mathcal{C} in “bands” numbered from the bottom to the top, in the following way.

- The 0-th band consists only of leaves
- Odd bands consist only of addition or multiplication gates.
- Even bands (other than 0) only consist of summation or production gates.
- The gates in band i can have edges incoming from only bands $j \leq i$.

Now, given a circuit $\tilde{\mathcal{C}}$ of size \tilde{s} with summation and production gates, we express it as a quantified monotone circuit of size $O(s)$ by inducting on the number of bands in it.

For the base case, when $\tilde{\mathcal{C}}$ has up to two bands, it is already a quantified monotone circuit.

In general, if $\tilde{\mathcal{C}}$ has $2b'$ bands, we look at the circuit formed by bands $2b'$ and $(2b' - 1)$ as a quantified monotone circuit; let its size be s . By induction, the *multi-output* circuit formed by the bands 0 to $2b' - 2$ can be expressed as a multi-output, quantified monotone circuit of size at most $s' = \tilde{s} - s$, call it \mathcal{C}' . Now from the hypothesis, the composition $\mathcal{C} \circ \mathcal{C}'$ is also computable by a quantified monotone circuit of size at most $s + s' \leq \tilde{s}$. ◀

C.4 Proof of Theorem 3.2

► **Theorem 3.2.** *If a polynomial family $\{f_n\}$ of degree $\text{poly}(n)$ is computable by monotone succinct ABPs of complexity $\text{poly}(n)$, then $\{f_n\} \in \text{mVNP}$.*

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

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

Proof. Let $\beta(\mathbf{u}, \mathbf{v}, \mathbf{x})$ be the *monotone* $(2r + n)$ -variate polynomial computed by the circuit B . Due to B being monotone, 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, $\deg_{\mathbf{x}}(\beta(\mathbf{a}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\bar{1}, \bar{1}, \mathbf{x}))$ for all \mathbf{a}, \mathbf{b} . Similarly, $\deg_{\mathbf{x}}(\beta(\mathbf{s}, \mathbf{b}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\mathbf{s}, \bar{1}, \mathbf{x}))$ and $\deg_{\mathbf{x}}(\beta(\mathbf{a}, \mathbf{t}, \mathbf{x})) \leq \deg_{\mathbf{x}}(\beta(\bar{1}, \mathbf{t}, \mathbf{x}))$ for all \mathbf{a}, \mathbf{b} . This shows that if $\ell > 1$, then

$$\deg(f) = \deg(\beta(\mathbf{s}, \bar{1}, \mathbf{x}) \cdot \beta(\bar{1}, \bar{1}, \mathbf{x})^{\ell-2} \cdot \beta(\bar{1}, \mathbf{t}, \mathbf{x})) \geq \ell - 2. \quad \triangleleft$$

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

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

This can be rewritten as follows.

$$\sum_{\mathbf{a}_1, \dots, \mathbf{a}_{d+1}} \left(2^{-r(d+1)} \beta(\mathbf{s}, \mathbf{t}, \mathbf{x}) + \sum_{j=1}^{d+1} 2^{-r(d+1-j)} \beta(\mathbf{s}, \mathbf{a}_1, \mathbf{x}) \left(\prod_{k=1}^{j-1} \beta(\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{x}) \right) \beta(\mathbf{a}_j, \mathbf{t}, \mathbf{x}) \right)$$

This is clearly a poly-sized exponential sum as $d = \text{poly}(n)$ and B is a monotone circuit of size $\text{poly}(n)$. ◀

C.5 Proof of Theorem 3.7

We introduce a new shorthand for this section. For a vector $\mathbf{a} = \{a_1, a_2, \dots, a_\ell\}$ and a number $k \leq \ell$, we use $\mathbf{a}[k]$ to denote the *prefix* vector $\{a_1, a_2, \dots, a_k\}$. With this new notation, we can express the last line of our toy example in Section 4 is as follows.

$$f(x) = \text{sum}_{y_1} \text{sum}_{y_2, *, y_3, ***} \left(\prod_{\mathbf{a} \in \{0,1\}^3} g(x, y_1, y_2, \mathbf{a}[1], y_3, \mathbf{a}[3], a_1, a_2, a_3) \right)$$

We are now ready to prove Theorem 3.7. We start by recalling the statement of the theorem.

11:22 Monotone Classes Beyond VNP

► **Theorem 3.7.** *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 monotone polynomial $A(\mathbf{w})$ such that,*

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

where $A(\mathbf{w})$ potentially has circuit 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 C.3.** Suppose $f(\mathbf{x})$ can be expressed as the following quantified circuit.

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

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.

$$f(\mathbf{x}) = \text{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{a}[:M_1], \mathbf{y}_3, \mathbf{a}[:M_2], \dots, \mathbf{y}_{k+1}, \mathbf{a}[:M_k], \mathbf{z} = \mathbf{a}) \right)$$

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

$$\mathbf{Y} = \bigcup_{\mathbf{a} \in \{0,1\}^{M_k}} (\mathbf{y}_1 \cup \mathbf{y}_2, \mathbf{a}[:M_1] \cup \cdots \cup \mathbf{y}_{k+1}, \mathbf{a}[:M_k]) \quad \triangleleft$$

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 bound the degree of f from below.

$$\begin{aligned} \deg(f) &= \deg_{\mathbf{x}} \left(\text{sum}_{\mathbf{Y}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{a}[:M_1], \dots, \mathbf{y}_{k+1}, \mathbf{a}[:M_k], \mathbf{z} = \mathbf{a}) \right) \right) \\ (g \text{ is monotone}) &= \deg_{\mathbf{x}} \left(\prod_{\mathbf{a} \in \{0,1\}^{M_k}} g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a}) \right) \\ &\geq \sum_{\mathbf{a} \in \{0,1\}^{M_k}} \deg(g(\mathbf{x}, \mathbf{1}, \mathbf{z} = \mathbf{a})) \end{aligned}$$

Therefore, since f has degree $d = \deg(f)$, 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 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 $\mathbf{Y}_1 := \bigcup_{\mathbf{a} \in \mathcal{A}} (\mathbf{y}_1 \cup \mathbf{y}_2, \mathbf{a}[:M_1] \cup \cdots \cup \mathbf{y}_{k+1}, \mathbf{a}[:M_k])$, 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 C.3 and finish the proof as follows, where \mathbf{y}_a refers to $(\mathbf{y}_1, \mathbf{y}_{2,a[:M_1]}, \dots, \mathbf{y}_{k+1,a[:M_k]})$.

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

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. \blacktriangleleft