Homomorphism Polynomials complete for VP[*]

Arnaud Durand[1], Meena Mahajan[2], Guillaume Malod[1], Nicolas de Rugy-Altherre [1], and Nitin Saurabh[2]

1 Univ Paris Diderot, Sorbonne Paris Cité, IMJ-PRG, UMR 7586 CNRS, Sorbonne Université, UPMC Univ Paris 06, F-75013, Paris, France.
{durand,malod,nderugy}@math.univ-paris-diderot.fr

2 The Institute of Mathematical Sciences, CIT Campus, Chennai 600113, India.
{meena,nitin}@imsc.res.in

Abstract

The VP versus VNP question, introduced by Valiant, is probably the most important open question in algebraic complexity theory. Thanks to completeness results, a variant of this question, VBP versus VNP, can be succinctly restated as asking whether the permanent of a generic matrix can be written as a determinant of a matrix of polynomially bounded size. Strikingly, this restatement does not mention any notion of computational model. To get a similar restatement for the original and more fundamental question, and also to better understand the class itself, we need a complete polynomial for VP. Ad hoc constructions yielding complete polynomials were known, but not natural examples in the vein of the determinant. We give here several variants of natural complete polynomials for VP, based on the notion of graph homomorphism polynomials.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases algebraic complexity, graph homomorphism, polynomials, VP, VNP, completeness

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

One of the most important open questions in algebraic complexity theory is to decide whether the classes VP and VNP are distinct. These classes, first defined by Valiant in [13, 12], are the algebraic analogues of the Boolean complexity classes P and NP, and separating them is essential for separating P from NP (at least non-uniformly and assuming the generalised Riemann Hypothesis, over the field $\mathbb{C}$, [3]). Valiant established that the family of polynomials computing the permanent is complete for VNP under a suitable notion of reduction which can be thought of as a very strong form of polynomial-size reduction. The leading open question of VP versus VNP is often phrased as the permanent versus the determinant, as the determinant is complete for VP. However, the hardness of the determinant for VP is under the more powerful quasi-polynomial-size reductions. Under polynomial reductions, the determinant is complete for the possibly smaller class VBP. This naturally raises the question of finding polynomials which are complete for VP under polynomial-size reductions. Ad hoc families of generic polynomials can be constructed that are VP-complete, but, surprisingly, there are no known natural polynomial families that are VP-complete. Since complete problems characterise complexity classes, the existence of natural complete problems lends

[*] This work was supported by IFCPAR/CEFIPRA Project 4702-1(A).

added legitimacy to the study of a class. The determinant and the permanent make the classes VBP, VNP interesting; analogously, what characterises VP?

Our results and techniques

In this paper, we provide the first instance of natural families of polynomials that (1) are defined independently of the circuit definition of VP, and (2) are VP-complete. The families we consider are families of homomorphism polynomials. Formal definitions appear in Section 2, but here is a brief description. For graphs G and H, a homomorphism from "source graph" G to "target graph" H is a map from $V(G)$ to $V(H)$ that preserves edges. If G and H are directed, a directed homomorphism must preserve directed edges. Additionally, if the vertices of G and H are coloured, a coloured homomorphism must also preserve colours. Placing distinct variables on the vertices (X variables) and edges (Y variables) of the target graph H, we can associate with each homomorphism from G to H a monomial built using these variables. The homomorphism polynomial associated with G and H is the sum of all such monomials. Various variants can be obtained by (1) summing only over homomorphisms of a certain type $\mathcal{H}$, e.g., directed, coloured, injective,... (2) setting non-negative weights α on the vertices of G and using these weights while defining the monomial associated with a homomorphism. Thus the general form of a homomorphism polynomial is $f_{G,H,\alpha,\mathcal{H}}(X,Y)$. We show that over fields of characteristic zero, with respect to constant-depth oracle reductions, the following natural settings, in order of increasing generality, give rise to VP-complete families (Theorem 19):

1. G is a balanced alternately-binary-unary tree with n leaves, with a marker gadget added to the root, and with edge directions chosen in a specific way; H is the complete directed graph on n^6 nodes; α is 1 everywhere; $\mathcal{H}$ is the set of directed homomorphisms.
2. G is an undirected balanced alternately-binary-unary tree with n leaves; H is the complete undirected graph on n^6 nodes; α is 1 everywhere; the vertices are coloured with 5 colours in a specific way; $\mathcal{H}$ is the set of coloured homomorphisms.
3. G is a balanced binary tree with n leaves; H is a complete graph on n^6 nodes; α is 1 for every right child in G and 0 elsewhere; $\mathcal{H}$ is the set of all homomorphisms from G to H.

There seems to be a trade-off between the ease of describing the source and target graphs and the use of weights α. The first family above does not use weights (α is 1 everywhere), but G needs a marker gadget on a naturally defined graph. The second family also does not use weights (α is 1 everywhere), but the colouring of H is described with reference to previously known universal circuits. The third family has very natural source and target graphs, but requires non-trivial α. Ideally, we should be able to show VP-completeness with G and H as in the third family and with trivial weights as in the first two families; our hardness proofs fall short of this. Note however that the weights we use are 0-1 valued. Such 0-1 weights are commonly used in the literature, see, e.g., [2].

A crucial ingredient in our hardness proofs is the fact that VP circuits can be depth-reduced [14] and made multiplicatively disjoint [8] so that all parse trees are isomorphic to balanced binary trees. Another crucial ingredient is that homogeneous components of a polynomial p can be computed in constant depth and polynomial size with oracle gates for p. The hardness proofs illustrate how the monomials in the generic VP-complete polynomial can be put in correspondence with a carefully chosen homogeneous component of the homomorphism polynomial (equivalently, with monomials associated with homomorphisms and satisfying some degree constraints in certain variables). Extracting the homogeneous component is what necessitates an oracle-reduction (constant depth suffices) for hardness. The coloured homomorphism polynomial is however hard even with respect to projections, the stricter form of polynomial-size reductions which is more common in this setting.

For all the above families, membership in VP is shown in a uniform way by showing that a more general homomorphism polynomial, where we additionally have a set of variables Z for each pair of nodes $V(G) \times V(H)$, is in VP, and that the above variants can be obtained from this general polynomial through projections. The generalisation allows us to partition the terms corresponding to $\mathcal{H}$ into groups based on where the root of G is mapped, factorise the sums within each group, and recurse. A crucial ingredient here is the powerful Baur-Strassen Lemma 3 ([1]) which says that for a polynomial p computed by a size s circuit, p and all its first-order derivatives can be simultaneously computed in size $O(s)$.

We also show that when G is a path (instead of a balanced binary tree), the homomorphism polynomial family is complete for VBP. On the other hand, using the generalised version with Z variables, and letting G, H be complete graphs, we get completeness for VNP.

Previous related results

As mentioned earlier, very little was previously known about VP-completeness. In [3], Bürgisser showed that a generic polynomial family constructed recursively while controlling the degree is complete for VP (Bürgisser showed something even more general; completeness for relativised VP). The construction directly follows a topological sort of a generic VP circuit. In [10] (see also [11]), Raz used the depth-reduction of [14] to show that a family of "universal circuits" is VP-complete; any VP computation can be embedded into it by appropriately setting the variables. Both these VP-complete families are thus directly obtained using the circuit definition / characterization of VP. In [9], Mengel described a way of associating polynomials with constraint satisfaction programs CSPs, and showed that for CSPs where all constraints are binary and the underlying constraint graph is a tree, these polynomials are in VP. Further, for each VP-polynomial, there is such a CSP giving rise to the same polynomial. This means that for the CSP corresponding to the generic VP polynomial or universal circuit, the associated polynomial is VP-complete. The unsatisfactory element here is that to describe the complete polynomial, one again has to fall back to the circuit definition of VP. Similarly, in [4], it is shown that tensor formulas can be computed in VP and can compute all polynomials in VP. Again, to put our hands on a specific VP-complete tensor formula, we need to fall back to the circuit characterisation of VP.

For VBP, on the other hand, there are natural known complete problems, most notably the determinant and iterated matrix multiplication.

A somewhat different homomorphism polynomial was studied in [5]; for a graph H, the monomials of the polynomial f_n^H encode the distinct graphs of size n that are homomorphic to H. The dichotomy result established there gives completeness for VNP or membership in Valiant's analogue of AC^0; it does not capture VP.

Finally, a considerable number of works have been done during the last years on the related subject of counting graph homomorphisms (but mostly in the non uniform settings — i.e., when the target graph is fixed — see [7]) or counting models of CSP and conjunctive queries with connections to VP-completeness (see [6]).

Organization of this paper

In Section 2, basic definitions and notations and previous results used are stated. In Section 3 we describe the hardness of various homomorphism polynomials for VP. Membership in VP is established in Section 4. Completeness for VBP and VNP is discussed in Section 5. Due to space limitations, detailed proofs had to be omitted.

2 Preliminaries and Notation

An arithmetic circuit is a directed acyclic graph with leaves labeled by variables or field elements, internal nodes (called gates) labeled by one of the field operations $+$ and $\times$, and designated output gates at which specific polynomials are computed in the obvious way. If every node has fan-out at most 1 (only one successor), then the circuit is a formula (the underlying graph is a tree). If at every node labeled $\times$, the subcircuits rooted at the children of the node are disjoint, then the circuit is said to be multiplicatively disjoint. Notions of size and depth are similar to that of classical Boolean circuits. For more details about arithmetic circuits, see for instance [11].

A family of polynomials $\{f_n(x_1, \ldots, x_{t(n)})\}$ is p-bounded if $t(n)$ and degree(f_n) are $n^{O(1)}$. A p-bounded family $\{f_n\}$ is in VP if a circuit family $\{C_n\}$ of size $s(n) \in n^{O(1)}$ computes it.

▶ **Proposition 1** ([14, 8]). *If $\{f_n\}$ is in* VP*, then $\{f_n\}$ can be computed by polynomial-size circuits of depth $O(\log n)$ where each $\times$ gate has fan-in at most 2. Furthermore, the circuits are multiplicatively disjoint.*

We say that $\{f_n\}$ is a p-projection of $\{g_n\}$ if there is an $m(n) \in n^{O(1)}$ such that each f_n can be obtained from $g_{m(n)}$ by setting each of the variables in $g_{m(n)}$ to a variable of f_n or to a field element.

A **constant-depth c-reduction** from $\{f_n\}$ to $\{g_n\}$, denoted $f \leq_c g$, is a polynomial-size constant-depth circuit family with $+$ and $\times$ gates and oracle gates for g, that computes f. (This is akin to AC^0-Turing reductions in the Boolean world.)

A family $\{D_n\}$ of **universal circuits** computing a polynomial family $\{p_n\}$ is described in [10, 11]. These circuits are universal in the sense that that every polynomial $f(X_1, \ldots, X_n)$ of degree d, computed by a circuit of size s, can be computed by a circuit Ψ such that the underlying graph of Ψ is the same as the graph of D_m, for $m \in \text{poly}(n, s, d)$. (In fact, f_n can be obtained as a projection of p_m.) With minor modifications to $\{D_n\}$ (simple padding with dummy gates, followed by the multiplicative disjointness transformation from [8]), we can show that there is a universal circuit family $\{C_n\}$ in the normal form described below:

▶ **Definition 2** (Normal Form Universal Circuits). A universal circuit $\{C_n\}$ in normal form is a circuit with the following structure:

- It is a layered and semi-unbounded circuit, where $\times$ gates have fan-in 2, whereas $+$ gates are unbounded.
- Gates are alternating, namely every child of a $\times$ gate is a $+$ gate and vice versa. Without loss of generality, the root is a $\times$ gate.
- All the input gates have fan-out 1 and they are at the same level, i.e., all paths from the root of the circuit to an input gate have the same length.
- C_n is a multiplicatively disjoint circuit.
- Input gates are labeled by distinct variables. In particular, there are no input gates labeled by a constant.
- Depth $(C_n) = 2k(n) = 2c\lceil\log n\rceil$; number of variables $(\bar{x}) = v_n$; and size $(C_n) = s_n$, which is polynomial in n.
- The degree of the polynomial computed by the universal circuit is n.

We will identify the directed graph of the circuit, where each edge e is labeled by a new variable X_e, by the circuit itself. Let $(\mathsf{f}_{C_n}(\bar{x}))_n$ be the polynomial family computed by the universal circuit family in normal form.

▶ **Lemma 3** ([1]). *Let $L(p_1, p_2, \ldots, p_k)$ denote the size of a smallest circuit computing the polynomials p_i at k of its nodes. For any $f \in \mathbb{F}[\bar{x}]$,*

$$L\left(f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right) \leq 3L\left(f\right).$$

The coefficient of a particular monomial in a polynomial can be extracted as described below. It appears to be folklore, and was noted in [3]; a version appears in [5] (Lemma 2).

▶ **Lemma 4** (Folklore). *Let F be any field of characteristic zero.*

1. *Let p be a polynomial in $F(\bar{W})$, with total degree at most D. Let m be any monomial, with k distinct variables appearing in it. The coefficient of m in p can be computed by a $O(k)$-depth circuit of size $O(Dk)$ with oracle gates for p.*
2. *Let p be a polynomial in $F(\bar{X}, \bar{W})$, with $|\bar{W}| = n$ and total degree in $\bar{W}$ at most D. Let p_d denote the component of p of total degree in $\bar{W}$ exactly d. Then p_d can be computed by a constant depth circuit of size $O(Dn)$ with $O(D)$ oracle gates for p.*

We use (u, v) to denote an undirected edge between u and v, and $\langle u, v\rangle$ to denote a directed edge from u to v.

▶ **Definition 5** (Homomorphisms). Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two undirected graphs. A homomorphism from G to H is a mapping $\phi : V(G) \to V(H)$ such that the image of an edge is an edge; i.e., for all $(u, v) \in E(G)$, $(\phi(u), \phi(v)) \in E(H)$.

If G, H are directed graphs, then a homomorphism only needs to satisfy for all $\langle u, v\rangle \in E(G)$, at least one of $\langle\phi(u), \phi(v)\rangle, \langle\phi(v), \phi(u)\rangle$ is in $E(H)$. But a directed homomorphism must satisfy for all $\langle u, v\rangle \in E(G)$, $\langle\phi(u), \phi(v)\rangle \in E(H)$.

If c_G, c_H are functions assigning colours to $V(G)$ and $V(H)$, then a coloured homomorphism must also satisfy, for all $u \in V(G)$, $c_G(u) = c_H(\phi(u))$.

▶ **Definition 6** (Homomorphism polynomials (see, e.g., [2])). Let G and H be undirected graphs; the definitions for the directed case are analogous. Consider the set of variables $X \cup Y$ where $X = \{X_u | u \in V(H)\}$ and $Y = \{Y_{uv} | (u, v) \in E(H)\}$. Let $\alpha : V(G) \mapsto \mathbb{N}$ be a labeling of vertices of G by non-negative integers. For each homomorphism ϕ from G to H we associate the monomial

$$mon(\phi) \triangleq \left(\prod_{u \in V(G)} X_{\phi(u)}^{\alpha(u)}\right)\left(\prod_{(u,v) \in E(G)} Y_{\phi(u),\phi(v)}\right)$$

Let $\mathcal{H}$ be a set of homomorphisms from G to H. The homomorphism polynomial $f_{G,H,\alpha,\mathcal{H}}$ is defined as follows:

$$f_{G,H,\alpha,\mathcal{H}}(X, Y) = \sum_{\phi \in \mathcal{H}} mon(\phi) = \sum_{\phi \in \mathcal{H}} \left(\prod_{u \in V(G)} X_{\phi(u)}^{\alpha(u)}\right)\left(\prod_{(u,v) \in E(G)} Y_{\phi(u),\phi(v)}\right)$$

Some sets of homomorphisms we consider are **InjDirHom**: injective directed homomorphisms, **InjHom**: injective homomorphisms, **DirHom**: directed homomorphisms, **ColHom**: coloured homomorphisms, **Hom**: all homomorphisms.

▶ **Definition 7** (Parse trees (see, e.g., [8])). The set of parse trees of a circuit C is defined by induction on its size:

- If C is of size 1, it has only one parse tree, itself.

- If the output gate of C is a $\times$ gate whose children are the gates α and β, the parse trees of C are obtained by taking a parse tree of C_α, a parse tree of a disjoint copy of C_β and the edges from α and β to the output gate.
- If the output of C is a $+$ gate, the parse trees of C are obtained by taking a parse tree of a subcircuit rooted at one of the children and the edge from the (chosen) child to the output gate.

Each parse tree T is associated with a monomial by computing the product of the values of the input gates. We denote this value by $mon(\mathsf{T})$.

▶ **Lemma 8** ([8]). *If C is a circuit computing a polynomial f, then $f(\bar{x}) = \sum_{\mathsf{T}} \mathrm{mon}(\mathsf{T})$, where the sum is over the set of parse trees, T, of C.*

▶ **Proposition 9** ([8]). *A circuit C is multiplicatively disjoint if and only if any parse tree of C is a subgraph of C. Furthermore, a subgraph T of C is a parse tree if the following conditions are met:*
- *T contains the output gate of C.*
- *If α is a multiplication gate in T having gates β and γ as children in C, then the edges $\langle\beta,\alpha\rangle$ and $\langle\gamma,\alpha\rangle$ also appear in T.*
- *If α is an addition gate in T, it has only one child in T.*
- *Only edges and gates obtained in this way belong to T.*

3 Lower Bounds: VP-hardness

Here we study the question of whether all families of polynomials in VP can be computed by homomorphism polynomials. Instantiating G, H and α to our liking we obtain a variety of homomorphism polynomials that are VP-hard. We describe them in increasing order of generalisation.

▶ **Definition 10.** Let AT_k be a directed balanced alternately-binary-unary tree with k leaves. Vertices on an odd layer have exactly two incoming edges whereas vertices on an even layer have exactly one incoming edge. The first layer has only one vertex called root, and the edges are directed from leaves towards the root.

▶ **Lemma 11.** *The parse trees of C_n, the universal circuit in normal form, are subgraphs of C_n and are isomorphic to AT_n.*

This observation suggests a way to capture monomial computations of the universal circuit via homomorphisms from AT_k into C_n.

Injective Directed Homomorphism

▶ **Proposition 12.** *Consider the homomorphism polynomial where*
- *$G := \mathsf{AT}_m$.*
- *H is the directed graph corresponding to the universal circuit in normal form C_m.*
- *$\mathcal{H} :=$ set of injective directed homomorphisms from G to H.*
- *α is 1 everywhere.*

Then, the family $(f_{\mathsf{AT}_m,H,\alpha,\textbf{\textit{InjDirHom}}}(\bar{X},\bar{Y}))_m$, where $m \in \mathbb{N}$, is VP-hard for projections.

Proof. (Sketch) We want to express the universal polynomial through a projection. We set all X variables at leaves to the corresponding variables in C_m, and all other X variables and all Y variables to 1. The idea is to show that elements in **InjDirHom** are in bijection with parse trees of C_m, and compute the same monomials. It is easy to see that for every parse tree of C_m, there is a $\phi \in$ **InjDirHom** with exactly this image. On the other hand, every $\phi \in$ **InjDirHom** must map the directed paths of length $2 \log m$ in G to directed paths in H; this fact can be used to show that its image is a parse tree of C_m. ◀

▶ Remark. The hardness proof above will work even if H is the complete directed graph on poly(m) nodes. In the projection, we can set the $\bar{Y}$ variables to values in $\{0, 1\}$ such that the edges with variables set to 1 together form the underlying graph of C_n.

If we follow the proof of the previous proposition and look at the image of a given homomorphism in layers, we notice that "direction"-respecting homomorphisms basically ensured that we never fold back (in the image). In particular, the mapping respect layers. Furthermore "injectivity" helped ensure that vertices within a layer are mapped distinctly. This raises an intriguing question: can we eliminate either assumption (*direction* or *injectivity*) and still prove VP-hardness? We answer this question positively, albeit under a stronger notion of reduction.

Injective Homomorphisms

Let AT^u_k be defined as the alternately-binary-unary tree AT_k, but with no directions on edges.

▶ **Proposition 13.** *Consider the homomorphism polynomial where*

- $G := \mathsf{AT}^u_m$.
- *H is a complete graph (undirected) on poly(m), say m^6, nodes.*
- *$\mathcal{H}$:= set of injective homomorphisms from G to H.*
- *α is 1 everywhere.*

Then, the family $(f_{\mathsf{AT}^u_m,H,\alpha,\textbf{\textit{InjHom}}}(\bar{X}, \bar{Y}))_m$ is VP*-hard for constant-depth c-reductions.*

Proof. (Sketch.) Again, we want to express the universal polynomial. Setting some Y variables to 0 values allows us to pick out C_m from H. To enforce directedness of the injective homomorphisms, we assign a special variable r on the edges emerging from the root, and a special variable ℓ on edges reaching the leaves. (The remaining Y variables are set to 1; the X variables are set as in Proposition 12.) Now the coefficient of $\ell^m r^2$ in f extracts exactly the contribution of injective directed homomorphisms, and this, by Proposition 12, is the universal polynomial. The desired coefficient can be extracted by a constant-depth c-reduction, as described in Lemma 4. ◀

Directed Homomorphisms

Consider the directed alternately-binary-unary-tree AT_k. For every vertex in an odd layer there are two incoming edges. Flip the direction of the right edge for every such vertex. Note that the edges coming into the unary vertices at even layers are unchanged. Also connect a path $t_1 \to t_2 \to \cdots \to t_s$ to the root by adding an edge $\langle t_s, root\rangle$. The vertices $t_1, \ldots, t_s$ are new vertices. Denote this modified alternately-binary-unary-tree by $\mathsf{AT}^d_{k,s}$.

▶ **Theorem 14.** *Consider the homomorphism polynomial where*

- $G := \mathsf{AT}^d_{m,s}$ *for sufficiently large s in poly(m), say $s = m^7$.*
- *H is a complete directed graph on poly(m), say m^6, nodes.*
- *$\mathcal{H}$:= set of directed homomorphisms from G to H.*
- *α is 1 everywhere.*

Then, the family $(f_{\mathsf{AT}^d_{m,s},H,\alpha,\textbf{\textit{DirHom}}}(\bar{X}, \bar{Y}))_m$ is VP*-hard for constant-depth c-reductions.*

Proof. (Sketch.) To compute the universal polynomial, we set some Y variables to 0 to pick out from H the circuit C_m with a tail at the root. We assign special variables r and t on the first and last edge of the tail, and variable ℓ on the edges entering leaves of C_m. The idea is to show that homomorphism monomials with degree exactly 1 in r and in t and degree exactly m in ℓ are in bijection with parse trees of C_m (and compute the same corresponding monomials). The length of the tail and the degree in r and t, ensure that any directed homomorphism maps the tail in G to the tail attached to C_m, so the root of the copy of AT_m inside G is mapped to the root of C_m. The degree constraint in ℓ ensures that the leaves of AT_m are mapped to the leaves of C_m, thus preserving layers. An inductive argument based on layer numbers, beginning from the root, and using the multiplicative disjointness of C_m, shows that the homomorphisms must also be injective. This gives the bijection. ◀

Coloured Homomorphisms

In all the above hardness proofs we restricted the set of homomorphisms to be *direction-respecting*, or *injective*, or both. Here we show another restriction, called *colour-respecting*, that gives a VP-hard polynomial. Recall that a homomorphism from a coloured graph to another coloured graph is *colour-respecting* if it preserves the colour class of vertices.

Consider the following colouring of AT_k^u with colours *brown, left, right, white* and *green.* The root of AT_k^u is coloured *brown*, leaves are coloured *green.* For every gate on an even layer, if it is the left (resp. right) child of its parent then colour it *left* (resp. *right*). Every gate on an odd layer, except the root, is coloured *white.* Denote this coloured alternately-binary-unary-tree as AT_k^c.

We define a circuit to be *properly* coloured if the root is coloured *brown*, leaves are coloured *green*, all multiplication gates but the root are coloured *white* and all addition gates are coloured *left* or *right* depending on whether they are left or right child respectively.

We obtain a *properly* coloured circuit from the universal circuit C_n as follows. For all addition gates in C_n we make two coloured copies, one coloured *left* and the other coloured *right.* We add edge connections as follows: for a multiplication gate we add an incoming edge to it from the *left* (resp. *right*) coloured copy of the left (resp. right) child, and for an addition gate the coloured gates are connected as the original gate in the circuit C_n.

We say that an undirected complete graph H on M nodes is *properly* coloured if, for all $s_n \leq M/2$, there is an embedding of the graph that underlies an s_n-sized *properly* coloured universal circuit, into H.

▶ **Theorem 15.** *Consider the homomorphism polynomial where*
- $G := \mathsf{AT}_m^c$.
- *H is a* properly *coloured complete graph (undirected) on poly(m), say m^6, nodes.*
- *$\mathcal{H}$:= set of coloured homomorphisms from G to H.*
- *α is 1 everywhere.*

Then, the family $(f_{\mathsf{AT}_m^c,H,\alpha,\boldsymbol{ColHom}}(\bar{X},\bar{Y}))_m$, where $m \in \mathbb{N}$, is VP*-hard for projections.*

Proof. (Sketch) As before, Y variables pick out C_m from H. The brown and green colours ensure that the root and the leaves of G are mapped to the root and leaves of C_m respectively. Injectivity follows from an argument similar to the one used for Theorem 14. ◀

The generic homomorphism polynomial gives us immense freedom in the choice of G, target graph H, weights α and the set of homomorphisms $\mathcal{H}$. Until now we used several modified graphs along with different restrictions on $\mathcal{H}$ to capture computations in the universal circuit. The question here is: can we get rid of restrictions on the set of homomorphisms? We provide a positive answer, using instead weights on the vertices of the source graph.

Homomorphism with weights

For k a power of 2, let T_k denote a complete (perfect) binary tree with k leaves.

▶ **Theorem 16.** *Consider the homomorphism polynomial where*

- $G := \mathsf{T}_m$.
- H *is a complete graph (undirected) on poly(m), say m^6, nodes.*
- $\mathcal{H}$:= *set of all homomorphisms from G to H.*
- *Define α such that,*

$$\alpha(u) = \begin{cases} 0 & u = \textit{root} \\ 1 & \textit{if } u \textit{ is the right child of it's parent} \\ 0 & \textit{otherwise} \end{cases}$$

Then, the family $(f_{\mathsf{T}_m,H,\alpha,\boldsymbol{Hom}}(\bar{X},\bar{Y}))_m$ is VP*-hard for constant-depth c-reductions.*

Proof. (Sketch) Since the source graph is complete binary trees, we first need to compact parse trees and get rid of the unary nodes (corresponding to $+$ gates). We construct from the universal circuit C_n a graph J_n that allows us to get rid of the alternating binary-unary parse tree structure while maintaining the property that the compacted "parse trees" are subgraphs of J_n. The graph J_n has two copies g_L and g_R of each $\times$ gate and input gate of C_n. It also has two children attached to each leaf node. The edges of J_n essentially shortcut the $+$ edges of C_n.

As before, we use Y variables to pick out J_n from H. We assign special variables w on edges from the root to a node g_R, and z on edges going from a non-root non-input node u to some right copy node g_R. For an input node g in the "left sub-graph" of J_n, the new left and right edges are assigned c_ℓ and x respectively, where x is the corresponding input label of g in C_n, and the node at the end of the x edge is assigned a special variable y. In the right sub-graph, variable c_r is used.

We show that homomorphisms whose monomials have degree 1 in w, $2^k - 2$ in z, 2^{k-1} each in c_ℓ and c_r, and 2^k in y are in bijection with compacted parse trees in J_n. The argument proceeds in stages: first show that the homomorphism is well-rooted (using the degree constraint on w, c_ℓ, c_r and the 0-1 weights in G), then show that it preserves layers (does not fold back) (using the degree constraint on c_ℓ, c_r and y), then show that it is injective within layers (using the degree constraint in z and the 0-1 weights in G). ◀

4 Upper Bounds: membership in VP

In this section we will show that most of the variants of the homomorphism polynomial considered in the previous section are also computable by polynomial size arithmetic circuits. That is, the homomorphism polynomials are VP-Complete. For sake of clarity we describe the membership of a generic homomorphism polynomial in VP in detail. Then we explain how to obtain various instantiations via projections.

We define a set of new variables $\bar{Z} := \{Z_{u,a} \mid u \in V(G) \text{ and } a \in V(H)\}$. Let us generalise the homomorphism polynomial $f_{G,H,\alpha,\mathcal{H}}$ as follows:

$$f_{G,H,\mathcal{H}}(\bar{Z},\bar{Y}) = \sum_{\phi\in\mathcal{H}} \left(\prod_{u\in V(G)} Z_{u,\phi(u)} \right) \left(\prod_{(u,v)\in E(G)} Y_{\phi(u),\phi(v)} \right).$$

Note that for a 0-1 valued α, we can easily obtain $f_{G,H,\alpha,\mathcal{H}}$ from our generic homomorphism polynomial $f_{G,H,\mathcal{H}}$ via substitution of $\bar{Z}$ variables, setting $Z_{u,a}$ to $X_a^{\alpha(u)}$. (If α can take any non-negative values, then we can still do the above substitution. We will need subcircuits computing appropriate powers of the $\bar{X}$ variables. The resulting circuit will still be poly-sized and hence in VP, provided the powers are not too large.)

▶ **Theorem 17.** *The family of homomorphism polynomials $(f_m) = f_{G_m,H_m,\textbf{Hom}}(\bar{Z},\bar{Y})$ where*

- *G_m is T_m, the complete balanced binary tree with $m = 2^k$ leaves,*
- *H_m is K_n, complete graph on $n = poly(m)$ nodes,*

is in VP.

Proof. (Sketch.) The idea is to group the homomorphisms based on where they send the root of G_m and its children, and to recursively compute sub-polynomials within each group. The sub-polynomials in a specific group will have a specific set of variables in all their monomials. Thus the group can be identified by suitably combining partial derivatives of the recursively constructed sub-polynomials. (Note: this is why we consider the generalised polynomial with $\bar{Z}$ instead of $\bar{X}$ and α. If for some u, $\alpha(u) = 0$, then we cannot use partial derivatives to force sending u to a specific vertex of H.) The partial derivatives themselves can be computed efficiently using Lemma 3. ◀

▶ Remark. In the above theorem and proof, if G_m is AT^u_m instead of T_m, essentially the same construction works. The grouping of homomorphisms should be based on the images of the root and its children and grandchildren as well.

If G_m and H_m have directions, again everything goes through the same way.

If we want to consider a restricted set $\mathcal{H}$ of homomorphisms **DirHom** or **ColHom** instead of all of **Hom**, again the same construction works. All we need is that $\mathcal{H}$ can be decomposed into independent parts with a local stitching-together operator. That is, whether ϕ belongs to $\mathcal{H}$ can be verified locally edge-by-edge and/or vertex-by-vertex, so that this can be built into the decomposition and the recursive construction.

From Theorem 17, the discussion preceding it and the remark following it, we have:

▶ **Corollary 18.** *The polynomial families from Proposition 12, Theorems 14, 15, and 16 are all in* VP*.*

▶ Remark. It is not clear how to get a similar upper bound for **InjDirHom** when the target graph is the complete directed graph (remark following Proposition 12), or for the family from Proposition 13. We need a way of enforcing that the recursive construction above respects injectivity. This is not a problem for Proposition 12, though, because there the target graph is the graph underlying a multiplicatively disjoint circuit. Injectivity at the root and its children and grandchildren can be checked locally; the recursion beyond that does not fold back because the homomorphisms are direction-preserving. The construction may not work if the target graph is the complete directed graph.

From Corollary 18, Proposition 12, and Theorems 14, 15 and 16, we get our main result:

▶ **Theorem 19.** *1. The polynomial families from Proposition 12 and Theorem 15 are complete for* VP *with respect to p-projections.*
2. The polynomial families from Theorems 14 and 16 are complete for VP *with respect to constant-depth c-reductions.*

5 Characterizing other complexity classes

We complement our result of VP-completeness by showing that appropriate modification of G can lead to VBP-complete and VNP-complete polynomial families.

VBP **Completeness**

VBP is the class of polynomials computed by polynomial-sized algebraic branching programs. These are layered directed graphs, with edges labeled by field constants or variables, and with a designated source node s and target node t. For any path ρ in G, the monomial $mon(\rho)$ is the product of the labels of all edges in ρ. For two nodes u, v, the polynomial p_{uv} sums $mon(\rho)$ for all paths ρ from u to v. The branching program computes the polynomial p_{st}.

A well-known polynomial family complete for VBP is the determinant of a generic matrix. A generic complete polynomial for VBP is the polynomial computed by an ABP with (1) a source node s, $m-1$ layers of m nodes each, and a target node t, (2) complete bipartite graphs between layers, and (3) distinct variables $\bar{x}$ on all edges. This is also the iterated matrix multiplication polynomial IMM. It is easy to see that st paths play the same role here as parse trees did in the multiplicatively disjoint circuits.

▶ **Theorem 20.** *Consider the homomorphism polynomial where*
- *G is a simple path on $m+1$ nodes, $(u_1, u_2, \ldots, u_{m+1})$.*
- *H is a complete graph (undirected) on m^2 nodes.*
- *$\mathcal{H}$:= set of all homomorphisms from G to H.*
- *Define α such that* $\alpha(u) = \begin{cases} 1 & u = u_1 \text{ or } u = u_{m+1} \\ 0 & \text{otherwise} \end{cases}$

Then, the family $(f_{G,H,\alpha,\boldsymbol{Hom}}(\bar{X}, \bar{Y}))_m$, where $m \in \mathbb{N}$, is complete for VBP *under c-reductions.*

Proof. (Sketch.) The hardness proof is very similar to that in Theorem 16. Lemma 3 (and hence Theorem 17) doesn't work for branching programs; however we show membership by direct construction of an ABP. ◀

VNP **Completeness**

▶ **Theorem 21.** *Consider the homomorphism polynomial where*
- *G is the complete graph (undirected) on m nodes.*
- *H is the complete graph (undirected) on m nodes.*
- *$\mathcal{H}$:= set of all homomorphisms from G to H.*
- *All $\bar{Y}$ variables are set to 1.*

Then, the family $(f_{G,H,\boldsymbol{Hom}}(\bar{Z}))_m$, where $m \in \mathbb{N}$, is complete for VNP *under p-projections.*

Proof. (Sketch.) This homomorphism polynomial is exactly the per_m polynomial. ◀

6 Conclusion

We have shown that several natural homomorphism polynomials are complete for the algebraic complexity class VP. Our results are summarised below.

Complexity	G	H	$\mathcal{H}$	polynomial type	reduction
VP-complete	AT_m	$C_{m^{O(1)}}$	**InjDirHom**	$\alpha = \underline{1}$	p-projections
	AT^d_m	$DK_{m^{O(1)}}$	**DirHom**	$\alpha = \underline{1}$	$O(1)$-depth c-reductions
	AT^c_m	coloured $K_{m^{O(1)}}$	**ColHom**	$\alpha = \underline{1}$	projections
	T^u_m	$K_{m^{O(1)}}$	**Hom**	0-1 valued	$O(1)$-depth
VBP-complete	Path_m	$K_{m^{O(1)}}$	**Hom**	0-1 valued	$O(1)$-depth
VNP-complete	K_m	K_m	**Hom**	generalised ($\bar{Z}$ variables)	p-projections

It would be interesting to show all the hardness results with respect to p-projections. It would also be very interesting to obtain completeness while allowing all homomorphisms on simple graphs and eliminating vertex weights. Another question is extending the completeness results of this paper to fields of characteristic other than zero.

Perhaps more importantly, it would be nice to get still more examples of natural VP-complete problems, preferably from different areas. The completeness of determinant or iterated matrix multiplication for VBP underlies the importance of linear algebra as a source of "efficient" computations. Finding natural VP-complete polynomials in some sense means finding computational techniques which are (believed to be) stronger than linear algebra.

References

1 Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317 – 330, 1983.

2 Christian Borgs, Jennifer T. Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In *Topics in Discrete Math*, pages 315–371. Springer, 2006.

3 P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000.

4 Florent Capelli, Arnaud Durand, and Stefan Mengel. The arithmetic complexity of tensor contractions. In *Symposium on Theoretical Aspects of Computer Science STACS*, volume 20 of *LIPIcs*, pages 365–376, 2013.

5 Nicolas de Rugy-Altherre. A dichotomy theorem for homomorphism polynomials. In *Mathematical Foundations of Computer Science 2012*, volume 7464 of *LNCS*, pages 308–322. Springer Berlin Heidelberg, 2012.

6 Arnaud Durand and Stefan Mengel. The complexity of weighted counting for acyclic conjunctive queries. *J. Comput. Syst. Sci.*, 80(1):277–296, 2014.

7 Martin E. Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013.

8 Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *Journal of Complexity*, 24(1):16–38, 2008.

9 Stefan Mengel. Characterizing arithmetic circuit classes by constraint satisfaction problems. In *Automata, Languages and Programming*, volume 6755 of *LNCS*, pages 700–711. Springer Berlin Heidelberg, 2011.

10 Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6:135–177, 2010.

11 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

12 L. G. Valiant. Reducibility by algebraic projections. In *Logic and Algorithmic: International Symposium in honour of Ernst Specker*, volume 30, pages 365–380. Monograph. de l'Enseign. Math., 1982.

13 Leslie G. Valiant. Completeness classes in algebra. In *Symposium on Theory of Computing STOC*, pages 249–261, 1979.

14 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.