Variants of Homomorphism Polynomials Complete for Algebraic Complexity Classes

PRASAD CHAUGULE and NUTAN LIMAYE, Indian Institute of Technology Bombay, India
ADITYA VARRE, École Polytechnique Fédérale de Lausanne, Switzerland

We present polynomial families complete for the well-studied algebraic complexity classes VF, VBP, VP, and VNP. The polynomial families are based on the homomorphism polynomials studied in the recent works of Durand et al. (2014) and Mahajan et al. (2018). We consider three different variants of graph homomorphisms, namely *injective homomorphisms*, *directed homomorphisms*, and *injective directed homomorphisms*, and obtain polynomial families complete for VF, VBP, VP, and VNP under each one of these. The polynomial families have the following properties:

- The polynomial families complete for VF, VBP, and VP are model independent, i.e., they do not use a particular instance of a formula, algebraic branching programs, or circuit for characterising VF, VBP, or VP, respectively.
- All the polynomial families are hard under p-projections.

CCS Concepts: • **Theory of computation** → *Algebraic complexity theory*; *Circuit complexity*; *Complexity classes;*

Additional Key Words and Phrases: Algebraic circuit complexity, directed homomorphism, injective homomorphism, injective and directed homomorphism

ACM Reference format:
Prasad Chaugule, Nutan Limaye, and Aditya Varre. 2021. Variants of Homomorphism Polynomials Complete for Algebraic Complexity Classes. *ACM Trans. Comput. Theory* 13, 4, Article 21 (August 2021), 26 pages.
https://doi.org/10.1145/3470858

1 INTRODUCTION

Valiant [16] initiated the systematic study of the complexity of algebraic computation. There are many interesting computational problems that have an algebraic flavour, for example, determinant, rank computation, and matrix multiplication. In fact, any problem related to these can be reformulated as a problem about computing a certain related polynomial. There are many other combinatorial problems, which do not prima facie have an algebraic flavour, but they can also be reduced to the problem of computing a certain polynomial.

Authors' addresses: P. Chaugule and N. Limaye, Indian Institute of Technology Bombay, Mumbai, India; emails: {prasad, nutan}@cse.iitb.ac.in; A. Varre, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland; email: aditya.varre@epfl.ch.

1942-3454/2021/08-ART21 $15.00
https://doi.org/10.1145/3470858

Valiant's work spurred the study of these and many other polynomials and led to a classification of these polynomials as *easy to compute* and *possibly hard to compute*. To talk about the ease or the hardness of computation, it is vital to first formalise the notion of a model of computation.

An *arithmetic circuit* is one such model of computation that has been well studied. An arithmetic circuit is a **directed acyclic graph (DAG)** whose in-degree 0 nodes are labelled with variables ($X = \{x_1, \ldots, x_n\}$) or field constants (from, some field, say, $\mathbb{F}$). All the other nodes are labelled with operators $+, \times$. Each such node computes a polynomial in a natural way. The circuit has an out-degree zero node, called the output gate. The circuit is said to compute the polynomial computed by its output gate. The size of the circuit is the number of gates in it.

Any multivariate polynomial $p(X) \in \mathbb{F}[x_1, \ldots, x_n]$ is said to be *tractable* if its degree is at most $\text{poly}(n)$ and there is a $\text{poly}(n)$ sized circuit computing it. The class of such tractable polynomial families is called VP.

Many other models of computation have been considered in the literature such as arithmetic formulas and **algebraic branching programs (ABPs)**. An *arithmetic formula* is a circuit in which the underlying DAG is a tree. The class of polynomial families computable by polynomial sized arithmetic formulas is called VF.

An *algebraic branching program* is a layered DAG, where the edges between two consecutive layers are labelled with variables from X or constants from $\mathbb{F}$. For any two nodes in this layered graph a directed path between them is said to compute the monomial obtained by multiplying the labels of the edges along that path. The ABP has two designated nodes, say, s and t. The polynomial computed by the ABP is a sum of all the monomials computed by all the paths from s to t. The size of an ABP is the number of nodes in the underlying DAG. The class of polynomial families computed by polynomial sized ABPs is called VBP.

Another important class of polynomials studied in the literature (and defined in Reference [16]) is VNP. It is known that VF $\subseteq$ VBP $\subseteq$ VP $\subseteq$ VNP. In Reference [16], it was shown that *the Permanent* polynomial is *complete*[1] for the class VNP.[2] It was also shown that the syntactic cousin of the Permanent polynomial, namely *the Determinant* polynomial, is in VP. However, the Determinant is not known to be complete for the class VP. In fact, for the longest time there were no natural polynomials that were known to be complete for VP.

Bürgisser [2] proposed a candidate VP-complete polynomial that was obtained by converting a generic polynomial sized circuit into a VP-hard polynomial (similarly to how the Circuit Value Problem is shown to be hard for P). Subsequently, Raz [14] gave a notion of a *universal circuit* and presented a VP-complete polynomial arising from the encoding of this circuit into a polynomial. More recently, Mengel [13] as well as Capelli et al. [3] proposed characterisations of polynomials computable in VP. In these and other related works, the VP-complete polynomial families were obtained using the structure of the underlying circuit. (See, for instance, Reference [6] and references therein for other related work.)

Let us consider a similar scenario in the Boolean setting. Here are two problems.

(1) $\{G = (V, E) \mid G \text{ has a hamiltonian cycle}\}$ and
(2) $\{\langle M, x, 1^t\rangle \mid M \text{ accepts } x \text{ in at most } t \text{ steps on at least one non-deterministic branch}\}$.

Both the problems are known to be NP-complete, but unlike the first problem, the second problem essentially codes the definition of NP into a decision problem. In that sense, the second problem is dependent on the model used to define NP, but the first one is independent of it. It is useful to have

[1] The hardness is shown with respect to p-projections. We will define them formally in Section 2.2.
[2] Valiant [16] raised the question of whether the Permanent is computable in VP. This question is equivalent to asking whether VP= VNP, which is the algebraic analogue of the P vs. NP question.

many problems like the first one that are NP-complete, as each such problem conveys a property of the class of NP-complete languages that is not conveyed by its definition.

In the Boolean world, the study of NP-complete problems was initiated by the influential works of Cook and Levin [4, 9]. Over the years, we have discovered thousands of NP-complete problems. Similarly, many natural problems have also been shown to be P-complete. See, for instance, Reference [8], which serves as a compendium of P-complete problems. Most of these problems are model independent.

In contrast, in the arithmetic world there is a paucity of circuit-description-independent VP-complete problems. Truly circuit-description-independent VP-complete polynomial families were introduced in the works of Durand et al. [6] and Mahajan et al. [11]. In this article, we extend their works by giving more such families of polynomials complete for VP. Along the way, we obtain such polynomial families complete for VF, VBP, and VNP as well.

At the core of our article are *homomorphism polynomials*, variants of which were introduced in Reference [6] and Reference [11]. (In fact, in References [5, 7], some variants of homomorphism polynomials were defined and they were studied in slightly different contexts.) Informally, a homomorphism polynomial is obtained by encoding a combinatorial problem of counting the number of homomorphisms from one graph to another as a polynomial. Say we have two graphs, the source G and the target graph H[3]; then the problem of counting the number of a certain set of homomorphisms, say, $\mathcal{H}$, from the graph G to H can be algebrised in many different ways. One such way is to represent the counting problem as the following polynomial:

$$f_{G,H,\mathcal{H}} = \sum_{\phi \in \mathcal{H}} \prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))},$$

where $Y = \{Y_{(a,b)} \mid (a,b) \in E(H)\}$ and $\mathcal{H}$ is a set of homomorphisms from G to H.[4]

Related Work

Here we will compare our work with other closely related works. We start by comparing the results and their relative strengths and weaknesses. We then compare and contrast the set of techniques used by us and by the previous papers.

Comparison of results. Our work essentially builds on the ideas defined and discussed in the works of References [6, 11]. Although Reference [11] and Reference [15] provide homomorphism polynomial families complete for all the important algebraic complexity classes, Reference [6] raises an interesting question, which remains unanswered even in Reference [11] and Reference [15]. The question is as follows: Do there exist homomorphism polynomial families complete for algebraic classes such as VP or VNP when $\mathcal{H}$ is restricted to only injective homomorphisms (or only directed homomorphisms)? We explore this direction and answer this question positively. This helps us to obtain a complete picture of the work initiated in Reference [6, 11].

In an attempt to find out well-studied/natural/well-understood graph classes such that the homomorphism polynomials defined over such graph classes are complete for complexity classes like VP, VBP, VNP, we consider three restricted sets of homomorphisms, namely injective homomorphisms (denoted as $\mathcal{IH}$), directed homomorphisms (denoted as $\mathcal{DH}$), and injective directed homomorphisms (denoted as $\mathcal{IDH}$). Naturally, when we consider $\mathcal{DH}$ or $\mathcal{IDH}$, we assume that G and H are directed graphs. We then design pairs of classes of graphs that help us obtain

[3]In the literature, the source and the target graph are sometimes also referred as the pattern and the host graph, respectively.
[4]Note that if we set all Y variables to 1, then this polynomial essentially counts the number of homomorphisms from G to H.

Table 1. Comparison between the Results in Our Work and Previous Works

	VP		VBP and VF	VNP
	c-reductions	**p-projections**	**p-projections**	**p-projections**
InjDirHom	–	[6], ✓	✓	✓
InjHom	–	✓	✓	✓
DirHom	[6]	✓	[6], ✓	✓
Hom	[6]	[11]	[11]	[6][15]

A cell containing the symbol ✓ represents the polynomial family designed in this article.

polynomials complete for the following complexity classes: VF, VBP, VP, and VNP. Like in References [6, 11], our polynomials are also model independent, i.e., the graph classes we use can be defined without knowing anything about the exact structure of the formula, ABP, or the circuit. We show the hardness in all the cases under more desirable p-projections. Table 1 gives the list of our results.

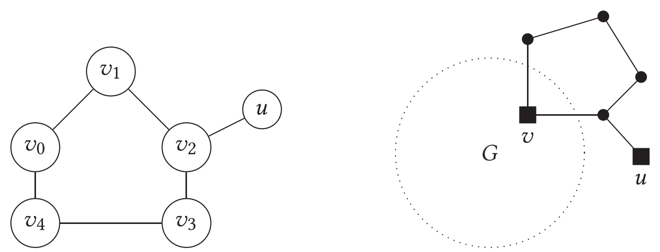
Comparison of techniques. There are a few different axes along which we can compare the previous and our proof ideas. (a) Type of reductions: c-reductions or p-projections (formally defined in Section 2.2). (b) The class of homomorphisms: all homomorphisms, injective homomorphisms, directed homomorphisms. (c) Two graph families are involved in all these works, that is, the source graph G and the target graph H. The kind of graphs used by these constructions. (d) The proof techniques involved in establishing the hardness and the upper bounds. The ideal situation is to obtain p-projections for all possible variants of homomorphisms while trying to keep graphs G and H as simple[5] as possible and the proof ideas as elementary as possible. While acknowledging that some of these notions are subjective, we compare the proof ideas below. We present the comparison in terms of undirected homomorphisms for the ease of discussion (i.e., we discuss injective homomorphisms and all homomorphisms and do not discuss the case of directed homomorphisms here).

Durand et al. [6] obtain a VP-hard polynomial family in the case of $\mathcal{H} = \mathcal{IH}$ very easily. They take G to be the *shape* of any parse tree of a universal circuit and H to be the undirected complete graph. They then show that the polynomial that sums over homomorphisms in $\mathcal{IH}$ between these two graph classes is VP-hard. The graph classes are very simple, but unfortunately, their hardness works only under the c-reductions. Moreover, the VP-upper bound of this family is not known.

In Reference [11], they do not consider $\mathcal{H} = \mathcal{IH}$. They instead come up with VP-complete families under p-projections when $\mathcal{H}$ is the class of all homomorphisms. Here, the graph G is a structure with small *tree-width*. However, graph H in their construction is a complete graph, which is very simple (to state). In their hardness proof as well as in their upper bound proof, they non-trivially use the fact that G has small tree-width.

For us, G is almost the same as in Reference [6], i.e., it is the shape of any parse tree of the universal circuit. The graph on the right-hand side is a novel graph structure, which we call a *Modified Block Tree*. This structure is essentially the graph underlying the universal circuit, where each node is copied several times and interconnections are modified appropriately. While this is more complicated than a complete graph, using the properties of this structure, we can bypass the use of bounded tree-width graphs (and their properties). We are also able to get an elementary

[5] By simple, we mean natural/well-understood/well-studied graphs.



(a) Spiked cycle $\mathcal{S}_5$, spiked node u. (b) $\mathcal{S}_5$ is attached to node v in G

Fig. 1. $\mathcal{S}_5$ attached to v in G. The distance between spiked node u and v is 2.

proof without using results such as the result of Baur and Strassen. This graph structure and some graph gadgets suffice to obtain the hardness as well as upper bound proofs.

Organisation. The rest of the article is organised as follows. We start with some notations and preliminaries in the following section. In Section 3, we present the details regarding VP-complete polynomial families. In Section 4, we present the results regarding VNP-complete polynomial families. In Section 5, we present the results regarding VBP and VF-complete polynomial families.

2 PRELIMINARIES

In this section, we introduce some notations and provide some preliminaries, which we will use in the rest of the article. For any integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any set S, we use $|S|$ to denote the cardinality of the set.

2.1 Graph Theoretic Notions

A cycle graph on n nodes, denoted as C_n, is a graph that has n nodes, say, $v_0, \ldots, v_{n-1}$, and n edges, namely $\{(v_{(i \mod n)}, v_{(i+1 \mod n)} \mid 0 \leq i \leq n-1\}$. We assume that the cycle graph is undirected unless stated otherwise. *A spiked cycle graph* on $n+1$ ($n \geq 3$) nodes, denoted as $\mathcal{S}_n$, is a cycle graph C_n with an additional edge (v, u), where u is an additional node that is not among $v_0, \ldots, v_{n-1}$, and $v \in \{v_0, \ldots, v_{n-1}\}$. We call the nodes $v_0, \ldots, v_{n-1}$ *the cycle nodes* and we call the additional node *a spiked*[6] *node.*

For a graph G, a cycle graph C_n is said to be attached to a node v of G, if one of the nodes of C_n is identified with the node v. A spiked cycle graph $\mathcal{S}_n$ is said to be attached to a node v of G if a node at distance 2 from the spiked node of $\mathcal{S}_n$ is identified with v (Figure 1).

2.2 Arithmetic Circuit Complexity Classes

Let X be a set of variables. Let $\mathbb{F}$ be any field of characteristic other than 2.[7]

An arithmetic circuit is a DAG in which the in-degree 0 nodes are called input gates, there is a unique out-degree 0 node called the output gate, all other nodes have in-degree 2 and are labelled with either + or ×. An input gate is typically labelled with a variable from X or a constant from $\mathbb{F}$. We define the polynomial computed by a circuit inductively. An input gate labelled x_i (or $c \in \mathbb{F}$) is said to compute the polynomial x_i (c, respectively). Let g be a gate with inputs g_1, g_2. Let $p_1(X)$, $p_2(X)$ be the polynomials computed by g_1, g_2, respectively. If g is labelled with × (or +), then the

[6] In the standard graph theory terminology, such a node is also called a pendant node.
[7] All our proofs go through for all characteristics, except the parts that involve the Permanent polynomial, which work for all characteristics other than 2.

polynomial computed by g is simply $p_1(X) \times p_2(X)$ (respectively, $p_1(X) + p_2(X)$). The polynomial computed by the circuit is the polynomial computed by the output gate. If the out-degree of every node in the circuit is 1, then it is called a formula.

The size of the circuit/formula is the number of nodes in the underlying graph. The depth of the circuit/formula is the length of the longest path from an input gate to the output gate.

Let $\{f_n(x_1, x_2, \ldots, x_{m(n)})\}_{n\in\mathbb{N}}$ be a family of polynomials. The family is said to be p-bounded if for each n the degree of f_n and $m(n)$ are polynomially bounded. A p-bounded family is said to be in VP (in VF) if, for each n, there is a circuit (respectively, formula) C_n such that C_n computes $f_n(x_1, \ldots, x_{m(n)})$ and the size of C_n, denoted as $s(n)$, is polynomially bounded.

An ABP is a layered directed graph $G = (V, E)$ such that the first layer contains a designated source node s and the last layer contains a designated sink node t. The edges are labelled with variables. For any s to t path ρ, we use f_ρ to denote the product of the variables labelling the edges in ρ. The polynomial computed by an ABP is $\sum_\rho f_\rho$, where ρ is an s to t path.

A p-bounded family is said to be in VBP if, for each n, there is an ABP P_n such that P_n computes $f_n(x_1, \ldots, x_{m(n)})$ and the size of P_n, denoted as $s(n)$, is at most $n^{O(1)}$.

Finally, a family of polynomials $\{f_n\}_{n\in\mathbb{N}}$ over $r(n)$ variables and of degree $d(n)$ is said to be in VNP if $r(n), d(n) \in n^{O(1)}$ and there is another polynomially bounded function $m(n)$ and a family of polynomials $\{g_n\}_{n\in\mathbb{N}}$ in VP such that for each $n \in \mathbb{N}$, g_n has $r(n) + m(n)$ variables denoted as $X = \{x_1, \ldots x_{r(n)}\}$, $Y = \{y_1, \ldots, y_{m(n)}\}$, and $f_n(X) = \sum_{y_1\in\{0,1\},\ldots,y_{m(n)}\in\{0,1\}} g_n(X, Y)$.

We now define p-projections and c-reductions.

Definition 2.1. A family of polynomials $\{f_n\}_{n\in\mathbb{N}}$ is a p-projection of another family of polynomials $\{g_n\}_{n\in\mathbb{N}}$ if there is a polynomially bounded function $m : \mathbb{N} \to \mathbb{N}$ such that for each $n \in \mathbb{N}$, f_n can be obtained from $g_{m(n)}$ by setting its variables to one of the variables of f_n or to field constants.

Definition 2.2. A family of polynomials $\{f_n\}_{n\in\mathbb{N}}$ is a c-reduction of another family of polynomials $\{g_n\}_{n\in\mathbb{N}}$ if there is a polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$ such that for each $n \in \mathbb{N}$, f_n can be computed by an arithmetic circuit of size poly (n) with $+$, $\times$ gates and oracle gates for $g_{t(n)}$.

2.3 Normal Form Circuits and Formulas

In this section, we present some other important notions regarding normal form circuits. We say that an arithmetic circuit is *multiplicatively disjoint* if the graphs corresponding to the subcircuits rooted at the children of any multiplication gate are node disjoint. We use a notion of a normal form of a circuit as defined in Reference [6]. We first define the notion of a universal circuit. This notion was defined in Reference [14] and was used in References [6, 11].

Definition 2.3 (Universal Circuit). A circuit D is said to be a (n, s, d)-universal circuit if for any polynomial f_n of degree d that can be computed by a size s circuit, there is another circuit Φ computing f_n such that the DAG underlying Φ is the same as that of D.

We assume that $s, d : \mathbb{N} \to \mathbb{N}$ are both functions of n. A family $\{D_n\}_{n\in\mathbb{N}}$ of $(n, s(n), d(n))$-universal circuits is defined in the usual way. If s, d are polynomially bounded functions of n, then we drop the parameters (n, s, d) from the description of the universal circuit.

Using the notion of universal circuits, we define the notion of the normal form for circuits.

Definition 2.4 ([6]). A family of universal circuits $\{D_n\}_{n\in\mathbb{N}}$ in the normal form is a family of circuits such that for each $n \in \mathbb{N}$, D_n has the following properties:

- It is a layered circuit in which each $\times$ gate ($+$ gate) has fan-in 2 (unbounded fan-in, respectively).

- Without loss of generality the output gate is a + gate. Moreover, the circuit has an alternating structure, i.e., the children of + (×) gates are × (+, respectively) gates, unless the children are in-degree 0 gates, in which case they are input gates.
- The input gates have out-degree 1. They all appear on the same layer, i.e., the length of any input gate to output gate path is the same.
- D_n is multiplicatively disjoint.
- Input gates are labelled by variables and no constants appear at the input gate.
- The depth of D_n is $2c\lceil \log n \rceil$, for some constant c. The number of variables, $v(n)$, and size of the circuit, $s(n)$, are both polynomial in n.
- The degree of the polynomial computed by the circuit is n.

We now recall a notion of a parse tree of a circuit from References [6, 12].

Definition 2.5 ([12]). The set of parse trees of a circuit C, $\mathcal{T}(C)$, is defined inductively based on the size of the circuit as follows.

- A circuit of size 1 has itself as its unique parse tree.
- If the circuit size is more than 1, then the output gate is either a × gate or a + gate.
  (i) if the output gate g of the circuit is a × gate with children g_1, g_2, and, say, C_{g_1}, C_{g_2} are the circuits rooted at g_1 and g_2, respectively, then the parse trees of C are obtained by taking a node disjoint copy of a parse tree of C_{g_1} and a parse tree of C_{g_2} along with the edges (g, g_1) and (g, g_2).
  (ii) if the output gate g of the circuit is a + gate, then the parse trees of C are obtained by taking a parse tree of any one of the children of g, say, h, and the edge (g, h).

It is easy to see that a parse tree computes a monomial. For a parse tree T, let f_T be the monomial computed by T. Given a circuit C (or a formula F), the polynomial computed by C (by F, respectively) is equal to $\sum_{T \in \mathcal{T}(C)} f_T$ ($\sum_{T \in \mathcal{T}(F)} f_T$, respectively).

We use the following fact about parse trees proved in Reference [12].

Proposition 2.6 ([12]). *A circuit C is multiplicatively disjoint if and only if every parse tree of C is a subgraph of C. Moreover, a subgraph T of C is a parse tree if:*

- *T contains the output gate of C.*
- *If g is a × gate in T, with children g_1, g_2 then the edges (g, g_1) and (g, g_2) appear in T.*
- *If g is a + gate in T, then it has a unique child in T, which is one of the children of g in C.*
- *No edges other than those added by the above steps belong to C.*

2.4 Graph Homomorphism, Its Variants, and Homomorphism Polynomials

We start with the definition of different variants of graph homomorphisms. Given two undirected graphs (the directed variant can be defined similarly) G and H, we say that $\phi : V(G) \to V(H)$ is *a homomorphism* from G to H if for any edge $(u, v) \in E(G)$, $(\phi(u), \phi(v)) \in E(H)$, i.e., the mapping preserves the edge relation. Note that, two different nodes in G can be mapped to the same node in H.

The homomorphism is said to be *an injective homomorphism* if additionally for any node $a \in V(H)$, $|\phi^{-1}(a)| \leq 1$, i.e., at most one node of G can be mapped to a node of H. Neither homomorphisms nor injective homomorphisms are required to be surjective, i.e., it is possible that $|V(G)| \leq |V(H)|$.

If the graphs G, H are directed, then the notion of homomorphism is modified to additionally preserve the directed edges. Formally, $\phi : V(G) \to V(H)$ is said to be *a directed homomorphism* from G to H if for any directed edge $(u, v) \in E(G)$, $(\phi(u), \phi(v))$ is a directed edge in $E(H)$.

Definition 2.7. Let G, H be two undirected graphs (or directed graphs). Let $Y = \{Y_{a,b} \mid (a,b) \in E(H)\}$ be a set of variables. Let $\mathcal{IH}$, $\mathcal{DH}$, $\mathcal{IDH}$ be a set of injective homomorphisms (in case of undirected graphs), directed homomorphisms (in case of directed graphs) and injective directed homomorphisms (in case of directed graphs) from G to H, respectively. Then the homomorphism polynomial $f_{G,H,\Delta}$ is defined as follows: $f_{G,H,\Delta}(Y) = \sum_{\phi \in \Delta} \prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))}$, where $\Delta \in \{\mathcal{IH}, \mathcal{DH}, \mathcal{IDH}\}$.

3 POLYNOMIAL FAMILIES COMPLETE FOR VP

Before going into the details of our results for class VP, we state the basic proof framework that is very similar to the proof framework in Reference [11] and Reference [6].

3.1 Basic Proof Framework for VP-completeness

(1) Consider a polynomial family, say, P_n, which is in VP and hence computed by the universal circuit in the normal form, say, D_n.
(2) We observe that every parse tree of D_n has the same shape and can be captured using a graph structure that we call "Alternating-Unary-Binary-tree."
(3) Like in Reference [11] and Reference [6], we use this observation and design our graph family G_n. As a result, for all our constructions, "Alternating-unary-binary tree" is essentially the main part of graph G_n.
(4) The next important part of our construction is designing the graph family H_n. We consider the circuit D_n and transform it into a more general circuit-independent graph structure, which we call "block tree." The "block tree" is constructed in such a way that it essentially simultaneously embeds all the parse trees of circuit D_n. The design of the block-tree is novel and was not used in the prior works.
(5) We further modify the "Alternating-unary-binary-tree" and convert it to G_n and modify the "Block tree" to graph H_n as needed, so that the polynomial P_n can be obtained as the projection of homomorphism polynomial $f_{G_n,H_n,\Delta}$, where $\Delta \in \{\mathcal{IH}, \mathcal{IDH}, \mathcal{DH}\}$.
(6) We then show the containment of $f_{G_n,H_n,\Delta}$ in VP using a simple inductive argument. This completes an overview of the proof idea.

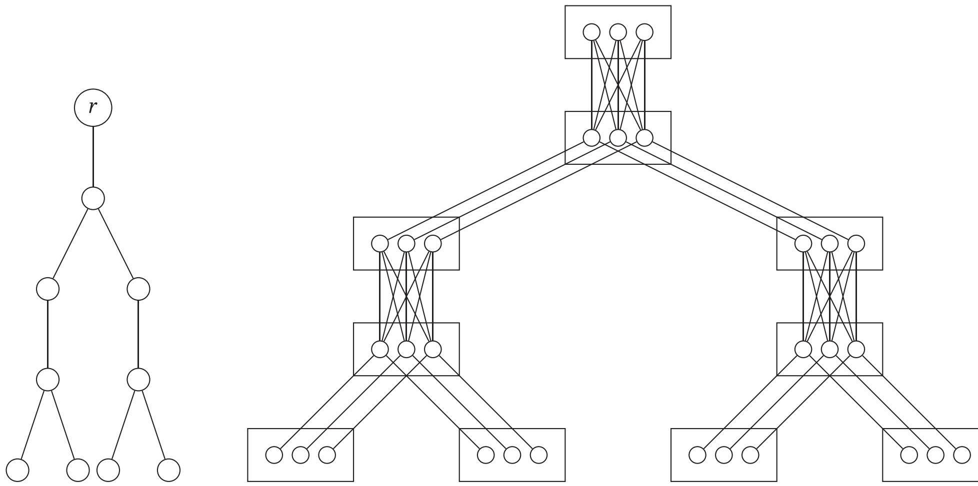
3.2 Injective Homomorphisms

We give some definitions of various graph classes.

Definition 3.1 (Balanced Alternating-Unary-Binary Tree). A balanced alternating-unary-binary tree with k layers, denoted as AT_k, is a layered tree in which the layers are numbered from $1, \ldots, k$, where the layer containing the root node is numbered 1 and the layer containing the leaves is numbered k. The nodes on an even layer have exactly two children and the nodes on an odd layer have exactly one child. Figure 2(a) shows an alternating-unary-binary tree with five layers.

Definition 3.2 (Block Tree). Let $\mathsf{BT}_{k,s}$ denote an alternately-unary-binary *block tree*, which is a graph obtained from AT_k by making the following modifications: Each node u of AT_k is converted into a block B_u consisting of s nodes. The block corresponding to the root node is called the root block. The blocks corresponding to the nodes on the even (odd) layers are called binary (unary, respectively) blocks. If v is a child of u in AT_k, then B_v is said to be a child of B_u in $\mathsf{BT}_{k,s}$.

After converting each node into a block of nodes, we add the following edges: say B is a unary block and block B' is its child, then for each node u in B and each node v in B' we add the edge (u,v). Moreover, if B is a binary block and B', B'' are its children, then we assume some ordering of the s nodes in these blocks. Say the nodes in B, B', B'' are $\{b_1, \ldots, b_s\}$, $\{b'_1, \ldots, b'_s\}$, and $\{b''_1, \ldots, b''_s\}$, respectively, then we add edges (b_i, b'_i) and (b_i, b''_i) for each $i \in [s]$.



(a) AT_5, with r as the root.

(b) $\mathsf{BT}_{5,3}$.

Fig. 2. Examples of AT_k and $\mathsf{BT}_{k,s}$.

Figure 2(b) shows a block tree $\mathsf{BT}_{k,s}$ where $k = 5$ and $s = 3$.

Let $k_1 = 3 < k_2 < k_3$ be three distinct fixed odd numbers such that $k_3 > k_2 + 2$.

Definition 3.3 (Modified-Alternating-Unary-Binary Tree). We attach a spiked cycle $\mathcal{S}_{k_1}$ to the root of AT_k. We attach a spiked cycle $\mathcal{S}_{j\times k_2}$ ($\mathcal{S}_{j\times k_3}$, respectively) to each left child node (right child node, respectively) in every odd layer $j > 1$. We call the graph thus obtained to be a modified alternating-unary-binary tree and denote it by MAT_k.

Definition 3.4 (Modified Block Tree). We start with $\mathsf{BT}_{k,s}$ and make the following modifications: we keep only one node in the root block and delete all the other nodes from the root block. We then attach a spiked cycle $\mathcal{S}_{k_1}$ to the only node in root block. We attach a spiked cycle $\mathcal{S}_{j\times k_2}$ ($\mathcal{S}_{j\times k_3}$, respectively) to each left child node (right child node, respectively) in every odd layer $j > 1$. We call the graph thus obtained a modified block tree and denote it by $\mathsf{MBT}_{k,s}$.

We identify each node in graphs MAT_k, $\mathsf{MBT}_{k,s}$ as either a *core node* or a *non core node*. We formally define this notion.

Definition 3.5 (Core Nodes and Non-Core Nodes). A *non-core node* is any node in MAT_k (or $\mathsf{MBT}_{k,s}$) that was not already present in AT_k (or $\mathsf{BT}_{k,s}$, respectively). Any node that is not a non-core node is a core node.

Consider the universal circuit family $\{D_n\}$ in normal form as in Definition 2.4. Let $m(n) = 2c\lceil \log n\rceil + 1$ be the number of layers in D_n and let $s(n)$ be its size. We prove the following theorem in this section.

Theorem 3.6. *The family $f_{G_n,H_n,\mathcal{IH}}(Y)$ is complete for class VP under p-projections, where G_n is $MAT_{m(n)}$ and H_n is $MBT_{m(n),s(n)}$.*

Informally, the main intuition of attaching spiked cycles to the graphs $\mathsf{AT}_{m(n)}$, $\mathsf{BT}_{m(n),s(n)}$ and converting it to $\mathsf{MAT}_{m(n)}$, $\mathsf{MBT}_{m(n),s(n)}$, respectively, is to ensure that certain nodes from the source graph G_n are mapped to certain nodes in the target graph H_n under all injective homomorphisms from G_n to H_n.

As the first step toward proving the theorem, we perform a few more updates to the normal form circuits we designed for polynomials in VP. Consider the universal circuit D_n in normal form as in Definition 2.4. We know that $m(n) = 2c\lceil \log n \rceil + 1$ is the number of layers in D_n and $s(n)$ is its size. From the definition of D_n, we know that any parse tree of D_n is isomorphic to $\mathsf{AT}_{m(n)}$. From such a circuit D_n, we construct another circuit D'_n, which has all the properties that D_n has and additionally the underlying graph of D'_n is a subgraph of the block tree $\mathsf{BT}_{m(n),s(n)}$, for $m(n), s(n)$ as mentioned above. Formally,

Lemma 3.7. *For every $n \in \mathbb{N}$, given any circuit D_n with $m(n) = 2c\lceil \log n \rceil + 1$ layers and size $s(n)$ in the normal form as in Definition 2.4, there is another circuit D'_n such that it has all the properties that the circuit D_n has and additionally it has the following properties:*

- *The polynomial computed by D'_n is the same as the polynomial computed by D_n.*
- *Every parse tree of D'_n is isomorphic to $AT_{m(n)}$.*
- *The underlying graph of D'_n is a subgraph of the block tree $BT_{m(n),s(n)}$.*
- *The size of D'_n is $\mathrm{poly}(s(n))$.*

Proof. To prove the lemma, we will give a construction of D'_n. Our construction will ensure that D'_n also has $m(n)$ layers. Let $u_1, u_2, \ldots, u_{t(j)}$ be the nodes in layer j of $\mathsf{AT}_{m(n)}$.[8] For each such node, we will have one block node in D'_n, i.e., we will add blocks $B_{u_1}, B_{u_2}, \ldots, B_{u_{t(j)}}$. We will say that a block B_u is a parent of a block B_v if u is a parent of v in $\mathsf{AT}_{m(n)}$. We will now describe the gates appearing in these blocks and then describe the connections between these blocks. That will complete the construction of D'_n.

Gates of D'_n: For j even, each block B_u in layer j has exactly one copy of every gate in layer j inside D_n. We know that for j even, the gates on the jth layer are $\times$ gates in D_n. Therefore, in D'_n too we only get $\times$ gates in the jth layer.

For j odd, each block B_u in layer j has $s(n)$ sub-blocks and each sub-block consists of a copy of each gate appearing in layer j inside D_n. All sub-blocks are identical in terms of the gates appearing in them. Note that all the gates appearing on the jth layer are $+$ gates by construction, if j is odd.

Wires of D'_n: Let g be a $\times$ gate in layer j inside a block B_u in D'_n. Say u has children u_1, u_2 in $\mathsf{AT}_{m(n)}$. (As g is a $\times$ gate, we know that j is even and hence that u is a binary node in $\mathsf{AT}_{m(n)}$.) Also, let $g = g_1 \times g_2$ in D_n and say among all the gates that g_1 (g_2, respectively) feeds into, g is the i_1th (i_2th, respectively) gate. We then add the following wires in D'_n: a wire from a copy of g_1 appearing in the i_1th sub-block of B_{u_1} to the gate g inside B_u and a wire from a copy of g_2 appearing in the i_2th sub-block of B_{u_2} to the gate g inside B_u.

Let g be a $+$ gate in layer j in the block B_u in D'_n. Say v is the child of u in $\mathsf{AT}_{m(n)}$. (As g is a $+$ gate, j is odd and u is a unary node in $\mathsf{AT}_{m(n)}$.) Say $g = g_1 + g_2 \ldots + g_k$ in D_n. Then we simply connect copies of $g_1, g_2, \ldots, g_k$ from B_v to the gate g in B_u.

Finally, we only keep one copy of the root gate in layer 1. If we assume that all the edges are directed from root toward the leaves, then we keep only edges induced by the nodes reachable from root in this directed graph. This finishes the construction of D'_n. It is easy to see that it has the properties stated in the lemma. ◻

From the construction of D'_n, we also get the following properties.

[8]Note that $t(j) = 2^{\lceil \frac{j}{2} \rceil - 1}$.

Proposition 3.8. *At most $s(n)$ copies of any + gate of D_n will appear in D'_n, where $s(n)$ is the size of D_n. Moreover, every copy of + gate in D'_n will be used at most once.*

We now prove Theorem 3.6 by first showing the hardness of the polynomial $f_{G_n,H_n,\mathcal{IH}}(Y)$ and then proving that it can be computed in VP.

3.2.1 VP hardness of $f_{G_n,H_n,\mathcal{IH}}(Y)$. We now show that if $f_n(X)$ is a polynomial computed in VP, then it is a p-projection of $f_{G_n,H_n,\mathcal{IH}}(Y)$. Let G_n, H_n be the source and target graphs defined in Theorem 3.6.

Let f_n be any polynomial in VP and D_n be the normal form universal circuit computing f_n with $m = 2c\lceil \log n \rceil + 1$ layers and size $s(n)$. We convert this circuit into $D^{'}_n$ as specified at the start of this section. As observed earlier, it still computes the polynomial computed by D_n. Let $\mathcal{G}'_n$ be the underlying graph of the circuit $D^{'}_n$. As $D^{'}_n$ is multiplicatively disjoint every parse tree of the circuit is a subgraph of $\mathcal{G}'_n$. Moreover, every parse tree is of the form $\mathsf{AT}_{m(n)}$.

If a spiked cycle is attached to a node v in layer ℓ of a layered graph, then we will say that all the nodes of the cycle belong to the same layer ℓ.

Let $\phi : G_n \to H_n$ be any injective homomorphism. Recall here G_n is $\mathsf{MAT}_{m(n)}$ and H_n is $\mathsf{MBT}_{m(n),s(n)}$. Let us use ϕ_i to denote the homomorphism ϕ restricted to layer i of G_n. Specifically, if V_i is the set of nodes in layer i of $\mathsf{MAT}_{m(n)}$, then ϕ_i is a homomorphism from V_i to the nodes of H_n. Let $\tilde{\phi_i}$ denote $\cup_{1\leq j\leq i}\phi_i$, i.e., the action of ϕ up to layer i. We will prove the following lemma inductively.

Lemma 3.9. *Let ϕ be an injective homomorphism from G_n to H_n. For any $i \in [m(n)]$, $\tilde{\phi_i}(G_n)$ is simply a copy[9] of the graph MAT_i inside H_n with the following additional properties:*

- *the root of MAT_i is mapped to the root of H_n.*
- *for any $i \in [m(n)]$, the core node u in layer i is mapped to a node in block B_u in layer i of H_n.*

Proof. The lemma can be proved easily using induction on $i \in [m(n)]$. We present all the details for the sake of completeness.

Base Case: For any injective homomorphism to survive, the root of G_n must be mapped to the root of H_n due to the presence of a spiked cycle graph $\mathcal{S}_{k_1}$ attached to the roots of both the graphs. This also satisfies the second property from the lemma, as the root of H_n is in B_r where r is the root of G_n.

Inductive case: We assume that the inductive hypothesis holds for all layers smaller than $i + 1$. Let u be a node in layer $i + 1$ of G_n. Let u_i be the parent of this node, which is in layer i. Say v_i is a node to which u_i is mapped in H_n. We break this case into two parts based on whether $i + 1$ is even or odd.

$i + 1$ is even and $i + 1 \geq 2$: Inductively we also have that the spiked cycle at node u_i is mapped injectively to the spiked cycle at v_i. Assume for the sake of contradiction that u does not get mapped to a node in layer $i + 1$. In this case, either u is mapped to a node in the spiked cycle attached to v_i or to some node in the layer $i - 1$ that is connected v_i. As the homomorphism is injective the first case cannot happen. The next case cannot happen as the outdegree of the gates in the odd layers is at most 1 and the neighbor of v_i in layer $i - 1$ will already have the parent of u_i mapped to it. Hence u must get mapped to a node in layer $i + 1$ that is adjacent to v_i thus to a node in B_u.

$i + 1$ is odd and $i + 1 \geq 3$: Let u_i and v_i be defined above. From the construction, u_i has two children, say, u, u', and v_i has two neighbours, say, v, v', in layer $i + 1$. Assume wlog that u is

[9] It is a layer preserving isomorphic copy that maps the root node of MAT_i to the root of H_n.

the left child of u_i. Hence, it has a spiked cycle $\mathcal{S}_{(i+1)\times k_2}$ attached to it. Similarly, if v is the left neighbour of v_i in the layer $i+1$, then it also has $\mathcal{S}_{(i+1)\times k_2}$ attached to it. As the size of the cycles attached to the nodes in layer $i-1$ is different, u cannot get mapped to a node in layer $i-1$. Hence u has to get mapped to v. This ensures that the cycle attached to u gets mapped to the cycle attached to v in an injective way. This completes the proof. □

We will now show that using this lemma we are done. We saw that $\mathcal{G}'_n$ is the subgraph of $\mathsf{BT}_{m,q}$, $m = 2c\lceil \log n\rceil + 1$ and $q = s(n)$ where it is embedded layer by layer.

We wish to set variables such that the monomial computed by each injective homomorphism is the same as the monomial computed by the corresponding parse tree. This can be achieved simply by setting variables as follows: Let e be an edge between two core nodes of H_n. If such an edge is not an edge in $\mathcal{G}'_n$, then set it to 0. (This carves out the graph $\mathcal{G}'_n$ inside H_n.) If such an edge is an edge associated with the leaf node, then locate the corresponding node in D'_n. It will be an input gate in D'_n. If the label of that input gate is x, then set this edge to x. If e is any other edge that appears in $\mathcal{G}'_n$, then set it to 1. (This allows for the circuit functionality to be realised along the edges of H_n.) Finally, suppose e is an edge between two non-core nodes (or between a core and a non-core node), i.e., along one of the attached cycles, then set it to 1. (This helps in suppressing the cycle edges in the final computation.)

This exactly computes the sum of all parse trees in the circuit D'_n, which shows that any polynomial computed in VP is also computed as a p-projection of $f_{G_n,H_n,\mathcal{IH}}(Y)$.

3.2.2 $f_{G_n,H_n,\mathcal{IH}}(Y)$ is in VP. The source graph G_n and target graph H_n are as described in the construction. We have already observed in Lemma 3.9 that all injective homomorphisms from G_n to H_n respect the layers. Therefore, it sufficies to compute only such layer respecting homomorphisms.

Construction of the circuit computing $f_{G_n,H_n,\mathcal{IH}}(Y)$. The construction of the circuit, say, C_n, is done from the bottom layer (i.e., from the leaves) to the top layer (i.e., to the root). For any *core node* $u \in V(\mathsf{MAT}_{m(n)})$ at layer ℓ of G_n and any *core node* a in block B_u at layer ℓ in H_n, we have a gate $\langle u, a\rangle$ in our circuit C_n at layer ℓ. Let us denote the sub-graph rooted at u in G_n by $G^{(u)}$ and that rooted at a in H_n to be $H^{(a)}$. Let $\mathcal{IH}_{(u,a)}$ be the set of injective homomorphism from sub-graph $G^{(u)}$ to $H^{(a)}$ where u is mapped to a. Let $f_{\langle u,a\rangle}$ be the polynomial computed at the gate $\langle u, a\rangle$.

We will describe the inductive construction of the circuit C_n starting with the leaves. We know that there is a spiked cycle $\mathcal{S}_{k_2\times m(n)}$ or $\mathcal{S}_{k_3\times m(n)}$ attached to each node at layer $m(n)$ in G_n.[10] For any spiked cycle $\mathcal{S}_k$ attached at a node x in H_n, let $\sigma^x_{\mathcal{S}_k}(Y)$ denote the monomial obtained by multiplying all the Y variables along the edges in $\mathcal{S}_k$ attached at x in H_n. Let u be a left (or right) child node in G_n at layer $m(n)$ and a be some node in B_u at layer $m(n)$ in H_n, then we set $\langle u, a\rangle = \sigma^a_{\mathcal{S}_{k_2\times m(n)}}(Y)$ (or $\langle u, a\rangle = \sigma^a_{\mathcal{S}_{k_3\times m(n)}}(Y)$, respectively).

Suppose we have a left (or right) child node, say, u, at layer i in G_n that has only one child, say, u' at layer $i+1$ in G_n. We know that there is a spiked cycle $\mathcal{S}_{k_2\times i}$ (or $\mathcal{S}_{k_3\times i}$, respectively) attached to u if it is the left child node (right child node, respectively). Let a be any node in B_u at layer i in H_n. Say a has t children, $a_1, \ldots, a_t$ in H_n. Inductively, we have gates $\langle u', a_\alpha\rangle$ for all $1 \le \alpha \le t$. We set

[10]Recall that $m(n) = 2c\lceil \log n\rceil + 1$, which is odd. Also this is without loss of generality.

$$\langle u, a\rangle = \sum_{\alpha=1}^{t} \langle u', a_\alpha\rangle \times Y_{(a,a_\alpha)} \times \sigma^a_{\mathcal{S}_{k_2\times i}}(Y), \quad \text{or} \tag{1}$$

$$\langle u, a\rangle = \sum_{\alpha=1}^{t} \langle u', a_\alpha\rangle \times Y_{(a,a_\alpha)} \times \sigma^a_{\mathcal{S}_{k_3\times i}}(Y), \tag{2}$$

depending on whether u is a left child or a right child of its parent in G_n, respectively. Suppose u in layer i in G_n has a left child u_1 and a right child u_2 in layer $i+1$. Let a be any node in the block B_u in H_n. Let a_1 and a_2 be the left child and right child of a in H_n, respectively. It is easy to see that a_1 resides in the block B_{u_1} in H_n and a_2 resides in the block B_{u_2} in H_n. Inductively, we have gates $\langle u_1, a_1\rangle$ and $\langle u_2, a_2\rangle$. We set

$$\langle u, a\rangle = \langle u_1, a_1\rangle \times Y_{(a,a_1)} \times \langle u_2, a_2\rangle \times Y_{(a,a_2)}. \tag{3}$$

This completes the description of C_n.

Correctness of C_n. Now to see the correctness of C_n, we prove the following lemma.

Lemma 3.10. *For any layer $i \in [m(n)]$, any node $u \in V(G_n)$ at layer i and any node a in block B_u also at layer i, $f_{\langle u,a\rangle}(Y) = \sum_{\phi\in\mathcal{IH}_{(u,a)}} \prod_{(u',u'')\in E(G^{(u)})} Y_{(\phi(u'),\phi(u''))}$, where $(\phi(u'),\phi(u'')) \in E(H^{(a)})$.*

See that when we invoke the above lemma for the roots of G_n and H_n, it proves that C_n computes the polynomial $f_{G_n,H_n,\mathcal{IH}}(Y)$.

Proof. We prove the lemma by induction on $i \in [m(n)]$.

Let u, a be the nodes as described in the statement of the lemma. We will use $\mathcal{M}_{u,a,\phi}(Y)$ as a short hand for the monomial $\prod_{(u',u'')\in E(G^{(u)})} Y_{(\phi(u'),\phi(u''))}$, where ϕ maps u to a.

For any *core node* $u \in V(\mathsf{MAT}_{m(n)})$ at layer ℓ of G_n and any *core node* a in block B_u at layer ℓ in H_n, if u is the left child (right child, respectively) of its parent in G_n then $f_{\langle u,a\rangle} = \sigma^a_{\mathcal{S}_{k_2\times m(n)}}(Y)$ (or $f_{\langle u,a\rangle} = \sigma^a_{\mathcal{S}_{k_3\times m(n)}}(Y)$, respectively). This is exactly what the construction does. Therefore, the base case holds. Now, assume that the lemma is true for layer $i+1$ then we prove that it is true for layer i. The proof proceeds in two cases.

Case 1 (u has one child): Suppose we have a left child node (or a right child node, respectively) u at layer i in G_n that has only one child, say, u', at layer $i+1$ in G_n. We know that there is a spiked cycle $\mathcal{S}_{k_2\times i}$ (or $\mathcal{S}_{k_3\times i}$, respectively) attached to u if it is the left child node (or right child node, respectively). Let a be any node in B_u at layer i in H_n. Suppose a has t children, say, $a_1, \ldots, a_t$ in H_n. Inductively, we have that $\langle u', a_\alpha\rangle$ computes the polynomial $f_{\langle u',a_\alpha\rangle}$ for all $1 \le \alpha \le t$. We construct $f_{\langle u,a\rangle}$ as follows:

$$\begin{aligned}
f_{\langle u,a\rangle} &= \sum_{\alpha=1}^{t} f_{\langle u',a_\alpha\rangle} \times Y_{(a,a_\alpha)} \times \sigma^a_{\mathcal{S}_{k_2\times i}}(Y) \\
&= \sum_{\alpha=1}^{t} \left(\left(\sum_{\phi_1\in\mathcal{IH}_{(u',a_\alpha)}} \mathcal{M}_{u',a_\alpha,\phi_1}(Y) \right) \times Y_{(a,a_\alpha)} \times \sigma^a_{\mathcal{S}_{k_2\times i}}(Y) \right) \\
&= \sum_{\alpha=1}^{t} \left(\sum_{\phi\in\mathcal{IH}_{(u,a)},\phi(u')=a_\alpha} \mathcal{M}_{u,a,\phi}(Y) \right) = \sum_{\phi\in\mathcal{IH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y)
\end{aligned}$$

or

$$
\begin{aligned}
f_{\langle u,a\rangle} &= \sum_{\alpha=1}^{t} f_{\langle u',a_\alpha\rangle} \times Y_{(a,a_\alpha)} \times \sigma^{a}_{\mathcal{S}_{k_3 \times i}}(Y) \\
&= \sum_{\alpha=1}^{t}\left(\left(\sum_{\phi_1 \in \mathcal{IH}_{(u',a_\alpha)}} \mathcal{M}_{u',a_\alpha,\phi_1}(Y)\right) \times Y_{(a,a_\alpha)} \times \sigma^{a}_{\mathcal{S}_{k_3 \times i}}(Y)\right) \\
&= \sum_{\alpha=1}^{t}\left(\sum_{\phi \in \mathcal{IH}_{(u,a)}, \phi(u')=a_\alpha} \mathcal{M}_{u,a,\phi}(Y)\right) = \sum_{\phi \in \mathcal{IH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y),
\end{aligned}
$$

depending on whether u is the left child or the right child of its parent in G_n, respectively. Here, the first equality comes from Equation (1) in the case when u is the left child and from Equation (2) when u is the right child. In both the cases, the second equality uses the inductive hypothesis. The third equality comes from simple rewriting of terms. Finally, the last equality follows from the fact that any homomorphism from node u of $\mathsf{MAT}_{m(n)}$ to a node a from the block B_u of $\mathsf{MBT}_{m(n),s(n)}$ must pick exactly one node from the block $B_{u'}$, where u' is a unique child of u in $\mathsf{MAT}_{m(n)}$. This ensures that only the injective homomorphisms from u to a propagate in the summation. This proves the inductive step in this case.

Case 2 (u has two children): Suppose u in layer i in G_n has a left child u_1 and a right child u_2 in layer $i+1$ in G_n. Let a be any node in block B_u in H_n. Let a_1 and a_2 be the left child and right child of a in H_n, respectively. It is easy to see that a_1 resides in block B_{u_1} in H_n and a_2 resides in block B_{u_2} in H_n. Inductively, we have gates $\langle u_1, a_1\rangle$ and $\langle u_2, a_2\rangle$ that computes the polynomial $f_{\langle u_1,a_1\rangle}$ and $f_{\langle u_2,a_2\rangle}$, respectively,

$$
\begin{aligned}
f_{\langle u,a\rangle} &= f_{\langle u_1,a_1\rangle} \times Y_{(a,a_1)} \times f_{\langle u_2,a_2\rangle} \times Y_{(a,a_2)} \\
&= \left(\left(\sum_{\phi_1 \in \mathcal{IH}_{(u_1,a_1)}} \mathcal{M}_{u_1,a_1,\phi_1}(Y)\right) \times \left(\sum_{\phi_2 \in \mathcal{IH}_{(u_2,a_2)}} \mathcal{M}_{u_2,a_2,\phi_2}(Y)\right) \times Y_{(a,a_1)} Y_{(a,a_2)}\right) \\
&= \sum_{\phi \in \mathcal{IH}_{(u,a)}, \phi(u_1)=a_1, \phi(u_2)=a_2} \mathcal{M}_{u,a,\phi}(Y) = \sum_{\phi \in \mathcal{IH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y).
\end{aligned}
$$

Here, the first equality comes from Equation (3), the second equality comes from the inductive hypothesis. The third equality is obtained by simple rewriting. Finally, the last equality is guaranteed by the fact that the subgraphs rooted at a_1 and a_2 do not share any node in common (as per the construction of H_n). Therefore, only the injective homomorphisms survive in the summation. This proves the inductive step in this case as well. This finishes the proof. □

3.3 Directed and Injective Directed Homomorphisms

We will give some definitions of various graph classes.

Definition 3.11 (Directed Balanced Alternating-Unary-Binary Tree). Let $\mathsf{AT}^{\mathsf{d}}_k$ denote the directed version of AT_k. The directions on the edges go from the root toward the leaves.

Definition 3.12 (Directed Block Tree). We use $\mathsf{BT}^{\mathsf{d}}_{k,s}$ to denote the directed version of $\mathsf{BT}_{k,s}$. The edges are directed from the root block toward the leaf blocks.

Let $k'_1 = 5 < k'_2 < k'_3 < k'_4$ be four distinct fixed natural numbers that are all pairwise co-primes.

Definition 3.13 (Modified Directed Alternating-Unary-Binary Tree). We attach a directed cycle $C_{k'_1}$ to the root of $\mathsf{AT}^{\mathsf{d}}_k$. We attach a directed cycle $C_{k'_2}$ to each node in every even layer in $\mathsf{AT}^{\mathsf{d}}_k$. We attach a directed cycle $C_{k'_3}$ ($C_{k'_4}$, respectively) to each *left child node* (*right child node*, respectively) in every odd layer (except the root node at layer 1) in $\mathsf{AT}^{\mathsf{d}}_k$. We call the graph thus obtained to be a modified directed alternating-unary-binary tree, $\mathsf{MAT}^{\mathsf{d}}_k$.

Definition 3.14 (Modified Directed Block Tree). We consider $\mathsf{BT}^{\mathsf{d}}_{k,s}$ and make the following modifications: we keep only one node in the root block node and delete all the other nodes from the root block node. We attach a directed cycle $C_{k'_1}$ to the only node in the root block of $\mathsf{BT}^{\mathsf{d}}_{k,s}$. We attach a directed cycle $C_{k'_2}$ to each node in every even layer in $\mathsf{BT}^{\mathsf{d}}_{k,s}$. We attach a directed cycle $C_{k'_3}$ ($C_{k'_4}$, respectively) to each *left child node* (*right child node*, respectively) in every odd layer (except the root node at layer 1) in $\mathsf{BT}^{\mathsf{d}}_{k,s}$. We call the graph thus obtained to be a modified directed block tree and denote it by $\mathsf{MBT}^{\mathsf{d}}_{k,s}$.

We identify each node in graphs $\mathsf{MAT}^{\mathsf{d}}_k$, $\mathsf{MBT}^{\mathsf{d}}_{k,s}$ as either a *core node* or a *non core node*. We formally define this notion.

Definition 3.15 (Core Nodes and Non-Core Nodes). A *non-core node* is any node in $\mathsf{MAT}^{\mathsf{d}}_k$ (or $\mathsf{MBT}^{\mathsf{d}}_{k,s}$) that was not already present in $\mathsf{AT}^{\mathsf{d}}_k$ (or $\mathsf{BT}^{\mathsf{d}}_{k,s}$, respectively). Any node that is not a non-core node is a core node.

We prove the following theorem in this section.

Theorem 3.16. *The families $f_{G_n,H_n,\mathcal{DH}}(Y)$, $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$ and $f_{G_n,H_n,\mathcal{IDH}}(Y)$ are complete for class VP under p-projections where G_n is $MAT^d_{m(n)}$, H_n is $MBT^d_{m(n),s(n)}$ and $K_{p(n)}$ is the complete graph obtained from H_n by adding all directed edges between every pair of nodes of H_n, where $p(n)$ is the number of nodes of H_n.*

Informally, the main intuition of attaching directed cycles to the graphs $\mathsf{AT}^{\mathsf{d}}_{m(n)}$, $\mathsf{BT}^{\mathsf{d}}_{m(n),s(n)}$ and converting it to $\mathsf{MAT}^{\mathsf{d}}_{m(n)}$, $\mathsf{MBT}^{\mathsf{d}}_{m(n),s(n)}$, respectively, is to ensure that certain nodes from the source graph G_n are mapped to certain nodes in the target graph H_n under all directed homomorphisms from G_n to H_n.

We first prove VP hardness of $f_{G_n,H_n,\mathcal{DH}}(Y)$ and $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$. We then show that these families are also contained in VP.

3.3.1 Hardness of $f_{G_n,H_n,\mathcal{DH}}(Y)$ and $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$. We first show that if $f_n(X)$ is a polynomial computed in VP, then it is a p-projection of $f_{G_n,H_n,\mathcal{DH}}(Y)$. This will prove the hardness of $f_{G_n,H_n,\mathcal{DH}}(Y)$. The hardness of $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$ will follow immediately from this by the following argument. As H_n is a subgraph of $K_{p(n)}$, a simple p-projection that retains all edges of H_n and sets all the other edges to 0 will give us the hardness.

The VP-hardness of $f_{G_n,H_n,\mathcal{DH}}(Y)$ can be proved using ideas similar to those used in proving the VP-hardness of $f_{G_n,H_n,\mathcal{IH}}(Y)$. The only difference is that here we attach directed cycles to the core-nodes of $\mathsf{AT}^{\mathsf{d}}_{m(n)}$ and $\mathsf{BT}^{\mathsf{d}}_{m(n)}$ instead of spiked cycles. The purpose of attaching the cycles is the same, i.e., to prohibit mappings that should not arise when $\mathcal{H} = \mathcal{DH}$.

Let $\phi : G_n \to H_n$ be any directed homomorphism. Let us use ϕ_i to denote the action of this homomorphism restricted to layer i on G_n. Let $\tilde{\phi}_i$ denote $\cup_{1\le j\le i}\phi_i$, i.e., the action of ϕ up to layer i. We will prove the following lemma inductively.

Lemma 3.17. *Let ϕ be a directed homomorphism from G_n to H_n. For any $i \in [m(n)]$, $\tilde{\phi}_i(G_n)$ is simply a copy of the graph MAT^d_i inside H_n with the additional properties that the root of MAT^d_i is mapped to the root of H_n and for any $i \in [m(n)]$, the core node u in layer i will be mapped to a node in block B_u in layer i of H_n.*

Proof. The proof of the lemma is similar to the proof of Lemma 3.9. The only difference here is that we can use the fact that we have directions on the edges.

Base case: Similarly to the base case of the proof of Lemma 3.9, here, too, it is easy to see that for any directed homomorphism to survive, the root of G_n must get mapped to the root of H_n. This is because we have a directed cycle of length k'_1 attached to the root of G_n as well as H_n and all the other core nodes have cycles of lengths that are coprime with respect to k'_1.

Inductive case: We assume that the inductive hypothesis holds for all layers smaller than $i+1$. We break this case into two parts based on whether $i+1$ is even or odd. The proofs for these cases are similar to the proofs of the corresponding cases in Lemma 3.9. Let u be a node in layer $i+1$ of G_n. Let u_i be the parent of this node, which is in layer i. Say v_i is a node to which u_i is mapped in H_n. Inductively we also have that the directed cycle attached at node u_i in G_n is mapped to the directed cycle attached at v_i in H_n.

$i+1$ **is even and** $i+1 \geq 2$**:** Suppose u is mapped to a node adjacent to v_i along the directed cycle $C_{k'_j}$ ($j=1$ or $j=3$ or $j=4$) attached at v_i in H_n. In this case, it is easy to see that the nodes in the cycle attached to u, namely $C_{k'_2}$, cannot be mapped along the cycle attached to v_i, given that k'_2 is coprime with respect to the length of the cycle attached to v_i. Hence the only possibility is that u gets mapped to one of the children of v_i in H_n, say, v. This is a good case. It is also easy to note that the directed cycle $C_{k'_2}$ attached at u in G_n must be mapped to the directed cycle $C_{k'_2}$ attached at v in H_n. Hence in this case we are done.

$i+1$ **is odd and** $i+1 \geq 3$**:** Note that u_i, v_i are as described above. Both u_i and v_i have two children each. Assume wlog that u is the left child of u_i. Either u gets mapped to the left child of v_i in H_n or to the right child of v_i in H_n or to the immediate next node to the v_i along the directed cycle $C_{k'_2}$ in H_n. As the order of the directed cycles attached to u, v_i and the right child of v_i are not compatible, it is easy to see that the last two cases listed above cannot happen. Therefore, the only node that u can get mapped to is the left child of v_i in H_n, say, v. Once again it is easy to see that the directed cycle $C_{k'_3}$ attached at u in G_n must get mapped to the directed cycle $C_{k'_3}$ attached at v in H_n. □

We will now show that using this lemma we are done. We consider the normal form circuit D'_n as designed in Section 3.2. Let $\mathcal{G}'_n$ be the underlying graph of D'_n. We direct the edges from the root to the leaves in this graph. Let the directed graph thus obtained be $\mathcal{G}''_n$. We know that $\mathcal{G}''_n$ is a subgraph of $\mathrm{BT}^{\mathrm{d}}_{m(n),s(n)}$ where it is embedded layer by layer.

We wish to set the variables such that the monomial computed by each directed homomorphism is the same as the monomial computed by the corresponding parse tree. This can be achieved simply by setting variables as follows: Let e be an edge between two core nodes of H_n. If such an edge is not an edge in $\mathcal{G}''_n$, then set it to 0. (This carves out the graph D'_n inside H_n.) If such an edge is an edge associated with the leaf node, then locate the corresponding node in D'_n. It will be an input gate in D'_n. If the label of that input gate is x, then set this edge to x. If e is any other edge that appears in $\mathcal{G}''_n$, then set it to 1. (This allows for the circuit functionality to be realised along the edges of H_n.) Finally, suppose e is an edge between two non-core nodes (or between a core and a non-core node), i.e., along one of the attached cycles, then set it to 1. (This helps in suppressing the cycle edges in the final computation.)

This exactly computes the sum of all parse trees in the circuit $D^{'}_n$, which shows that the any polynomial computed in VP is also computed as a p-projection of $f_{G_n,H_n,\mathcal{DH}}(Y)$.

3.3.2 $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$ and $f_{G_n,H_n,\mathcal{DH}}(Y)$ are in VP. In this section, we will show that $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$ is computable in VP. As noted above, $f_{G_n,H_n,\mathcal{DH}}(Y)$ reduces to $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$

under p-projections. Therefore, this will also show a VP upper bound for $f_{G_n,H_n,\mathcal{DH}}(Y)$. Let us denote the subgraph rooted at any core node u in G_n by $G^{(u)}$. Let $\mathcal{DH}_{(u,a)}$ be the set of directed homomorphisms from subgraph $G^{(u)}$ to $K_{p(n)}$ where u is mapped to node a in $K_{p(n)}$. The construction of the polynomial sized circuit for $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$ is fairly straightforward. We give the details for the sake of completeness.

Let u be a core node in G_n and a be any node in $K_{p(n)}$. We define the polynomial $\sigma^{u,a}(Y)$ as follows:

$$\sigma^{u,a}(Y) = \sum_{\phi \in \mathcal{DH}_{(u,a)}} \prod_{(u',v') \in E(C_u)} Y_{(\phi(u'),\phi(v'))},$$

where C_u is the directed cycle attached at the core node u. Basically, $\sigma^{u,a}$ is the polynomial that sums the monomials corresponding to all possible directed homomorphisms that map the cycle C_u to a subset of nodes in $K_{p(n)}$, while maintaining the mapping of u in G_n to a in $K_{p(n)}$. It is easy to see that if the number of nodes in the cycle C_u is k then the above circuit has size at most $O(p(n)^k)$. By our construction we know that the cycles attached at any core node in G_n are of constant size and $p(n)$ is polynomially bounded. Thus, for any u, a, we can compute $\sigma^{u,a}$ explicitly in polynomial size. We compute all such polynomials for every core node u in G_n and every node a in $K_{p(n)}$.

We give this proof in two steps. We first give the construction of the circuit C_n and then prove its correctness. We will build the circuit C_n inductively from bottom-most layer to the top-most layer of graph G_n.

Base Case: For any core node u in the bottom most layer of G_n and any node a in $K_{p(n)}$, we define a gate $\langle u, a\rangle$. We set $\langle u, a\rangle = \sigma^{u,a}(Y)$.

Inductive Case: Suppose we have a core node u, at layer i in G_n that has only one core node as a child, say, u', at layer $i+1$ in G_n. Let a be any node in $K_{p(n)}$. Inductively, we have gates $\langle u', a'\rangle$ for all $a' \in V(K_{p(n)})$. We set

$$\langle u, a\rangle = \sum_{(a,a') \in E(K_{p(n)})} \langle u', a'\rangle \times Y_{(a,a')} \times \sigma^{u,a}(Y). \tag{4}$$

Suppose u in layer i in G_n has a left child u_1 and a right child u_2 in layer $i+1$ in G_n. Let a be any node in $K_{p(n)}$. Let a_1 and a_2 be any two neighbors of a. Inductively, we have gates $\langle u_1, a_1\rangle$ and $\langle u_2, a_2\rangle$. We set

$$\langle u, a\rangle = \sum_{(a,a_1),(a,a_2) \in E(K_{p(n)})} \langle u_1, a_1\rangle \times Y_{(a,a_1)} \times \langle u_2, a_2\rangle \times Y_{(a,a_2)} \times \sigma^{u,a}(Y). \tag{5}$$

Let OUT denotes the output gate of C_n. We set $OUT = \sum_{a \in V(K_{p(n)})} \langle r, a\rangle$ where r is the root node of graph G_n. This completes the description of the circuit C_n.

Let $f_{\langle u,v\rangle}$ denote the polynomial computed by gate $\langle u, v\rangle$ in C_n. To prove the correctness of circuit C_n, it is sufficient to prove the following lemma.

Lemma 3.18. *For any layer $i \in [m(n)]$, any node $u \in V(G_n)$ at layer i and any node a in $K_{p(n)}$,*

$$f_{\langle u,a\rangle}(Y) = \sum_{\phi \in \mathcal{DH}_{(u,a)}} \prod_{(u',u'') \in E(G^{(u)})} Y_{(\phi(u'),\phi(u''))},$$

where $(\phi(u'), \phi(u'')) \in E(K_{p(n)})$.

Let us assume that we invoke the above lemma for the root of G_n and any node a in $K_{p(n)}$. Then it immediately follows that the circuit C_n computes the polynomial $f_{G_n,K_{p(n)},\mathcal{DH}}(Y)$. The proof of the lemma is almost the same as the proof of Lemma 3.10.

We will use $\mathcal{M}_{u,a,\phi}(Y)$ as a short-hand for the monomial $\prod_{(u',u'')\in E(G^{(u)})} Y_{(\phi(u'),\phi(u''))}$, where ϕ is such that it maps u to a.

Proof. Base case: For any core node u in the bottom most layer in G_n and any node a in $K_{p(n)}$, $f_{\langle u,a\rangle} = \sigma^{u,a}(Y)$, which is exactly equal to polynomial $\sum_{\phi\in\mathcal{DH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y)$. Therefore, the base case holds.

Inductive case:
Case 1: Consider a unary core node u in G_n. Let u' be its only child in G_n. We have

$$\begin{aligned}
f_{\langle u,a\rangle} &= \sum_{(a,a')\in E(K_{p(n)})} f_{\langle u',a'\rangle} \times Y_{(a,a')} \times \sigma^{u,a}(Y)\\
&= \sum_{(a,a')\in E(K_{p(n)})} \left(\left(\sum_{\phi\in\mathcal{DH}_{(u',a')}} \mathcal{M}_{u',a',\phi}(Y)\right) \times Y_{(a,a')} \times \sigma^{u,a}(Y)\right)\\
&= \sum_{(a,a')\in E(K_{p(n)})} \left(\sum_{\substack{\phi\in\mathcal{DH}_{(u,a)}\\ \phi(u')=a'}} \mathcal{M}_{u,a,\phi}(Y)\right) = \sum_{\phi\in\mathcal{DH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y).
\end{aligned}$$

Here the first equality follows from Equation (4), and the second equality follows from the inductive hypothesis step. The third equality is obtained just by rewriting. The fourth equality follows from the fact that the set of directed homomorphism ϕ can always be partitioned into sets depending on where ϕ maps any node u' to.

Case 2: Consider a binary core node u in graph G_n. Let u_1 and u_2 be its left and right child, respectively, in graph G_n. We have

$$\begin{aligned}
f_{\langle u,a\rangle} &= \sum_{\substack{(a,a_1)\in E(K_{p(n)})\\ (a,a_2)\in E(K_{p(n)})}} f_{\langle u_1,a_1\rangle} \times Y_{(a,a_1)} \times f_{\langle u_2,a_2\rangle} \times Y_{(a,a_2)} \times \sigma^{u,a}(Y)\\
&= \sum_{\substack{(a,a_1)\in E(K_{p(n)})\\ (a,a_2)\in E(K_{p(n)})}} \left(\sum_{\phi\in\mathcal{DH}_{(u_1,a_1)}} \mathcal{M}_{u_1,a_1,\phi}(Y)\right) Y_{(a,a_1)}\\
&\quad \times \left(\sum_{\phi\in\mathcal{DH}_{(u_2,a_2)}} \mathcal{M}_{u_2,a_2,\phi}(Y)\right) Y_{(a,a_2)} \times \sigma^{u,a}(Y)\\
&= \sum_{\substack{(a,a_1)\in E(K_{p(n)})\\ (a,a_2)\in E(K_{p(n)})}} \left(\sum_{\substack{\phi\in\mathcal{DH}_{(u,a)}\\ \phi(u_1)=a_1\\ \phi(u_2)=a_2}} \mathcal{M}_{u,a,\phi}(Y)\right) = \sum_{\phi\in\mathcal{DH}_{(u,a)}} \mathcal{M}_{u,a,\phi}(Y).
\end{aligned}$$

Here the first equality follows from Equation (5), and the second equality follows from the inductive hypothesis step. The third equality is obtained just by rewriting. The fourth equality follows from the fact that for any binary core node u in G_n with children u_1 and u_2, the set of directed homomorphism ϕ can be partitioned into sets depending on where ϕ maps u_1 and u_2 to.

By the way we have constructed the circuit, finally we have, $f_{OUT} = \sum_{a\in V(K_{p(n)})} f_{(r,a)}$ where r is the root node of graph G_n. □

Observation. *From our construction of the polynomial, it is interesting to note that for any* $\phi \in \mathcal{DH}$,

- *a core node* u *in* G_n *must get mapped to some node* v *in the block* B_u *in* H_n. *That is, any two core-nodes in* G_n *must get mapped to two distinct core-nodes in* H_n.
- *Let a core-node* u *in* G_n *gets mapped to a core-node* v *in* H_n. *Let* C_u *and* C_v *denotes the directed cycles attached at nodes* u *(in* G_n*) and* v *(in* H_n*), respectively. It is clear that* C_u *must get exactly mapped to* C_v *in an injective way, where* $\phi(u) = v$.

This implies that every ϕ in $\mathcal{DH}$ is also injective. Therefore, in fact $\mathcal{DH} = \mathcal{IDH}$, where $\mathcal{IDH}$ is a set of all injective directed homomorphisms.

The observation shows that $f_{G_n,H_n,\mathcal{DH}}(Y)$ we constructed is in fact also $f_{G_n,H_n,\mathcal{IDH}}(Y)$.

Remark. We get a VP-complete polynomial family when the right-hand-side graph is a complete graph and $\mathcal{H} = \mathcal{DH}$. It would be interesting to get this feature even when $\mathcal{H} = \mathcal{IH}$ or $\mathcal{IDH}$. In the case of $\mathcal{H} = \mathcal{IH}$ or $\mathcal{IDH}$, although the VP-hardness of these families goes through, the containment of these families in VP is not straightforward (as in case of $\mathcal{H} = \mathcal{DH}$). It is worth noting however that all our constructions ensure that the graphs are model independent in all three cases, i.e., when $\mathcal{H}$ equals $\mathcal{IH}, \mathcal{DH}$, or $\mathcal{IDH}$.

4 POLYNOMIAL FAMILIES COMPLETE FOR VNP

In this section, we present VNP-complete polynomial families. At the core of our VNP-complete polynomials lies the Permanent polynomial, which is defined as follows:

$$Perm_n(X) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i \in [n]} x_{i,\sigma(i)},$$

where $X = \{x_{i,j} \mid i, j \in [n]\}$ and $\mathcal{S}_n$ is a set of all permutations on n elements. It is known that Permanent polynomial is complete for VNP.

4.1 Injective Homomorphisms

In this section, we present a polynomial family that is complete for VNP, when $\mathcal{H}$ is the class of injective homomorphisms.

4.1.1 Construction. Let $\widehat{G_n}$ be a bipartite graph with node partitions $V_1(\widehat{G_n})$ and $V_2(\widehat{G_n})$. Let $V_1(\widehat{G_n}) = \{u_i | 1 \leq i \leq n\}$ and $V_2(\widehat{G_n}) = \{v_i | 1 \leq i \leq n\}$. Let $E(\widehat{G_n}) = \{(u_i, v_i) | 1 \leq i \leq n\}$. We will do some modifications to $\widehat{G_n}$. Let $k_1 = 3 < k_2 \ldots < k_n < k_{n+1}$ be $n+1$ consecutive odd numbers. We attach a spiked cycle $\mathcal{S}_{k_i}$ to node u_i for all $1 \leq i \leq n$. We attach a spiked cycle $\mathcal{S}_{k_{n+1}}$ to node v_i for all $1 \leq i \leq n$. We call this modified version of $\widehat{G_n}$ as G_n.

Let $\widehat{H_n}$ be a bipartite graph with node partitions $V_1(\widehat{H_n})$ and $V_2(\widehat{H_n})$. Let $V_1(\widehat{H_n}) = \{u'_i | 1 \leq i \leq n\}$ and $V_2(\widehat{H_n}) = \{v'_i | 1 \leq i \leq n\}$. Let $E(\widehat{H_n}) = \{(u'_i, v'_j) | 1 \leq i, j \leq n\}$. We attach a spiked cycle $\mathcal{S}_{k_i}$ to node u'_i for all $1 \leq i \leq n$. We attach a spiked cycle $\mathcal{S}_{k_{n+1}}$ to node v'_i for all $1 \leq i \leq n$. We call this modified version of $\widehat{H_n}$ as H_n.

For this choice of G_n, H_n and $\mathcal{IH}$, we define the homomorphism polynomial, $f_{G_n,H_n,\mathcal{IH}}(Y)$ where $n \in \mathbb{N}$. Figure 3 and Figure 4 show the graphs G_n and H_n for $n = 3$, respectively.

Theorem 4.1. *The family* $f_{G_n,H_n,\mathcal{IH}}(Y)$ *is complete for class VNP under p-projections where* G_n *and* H_n *are as described above.*

4.1.2 Hardness of $f_{G_n,H_n,\mathcal{IH}}(Y)$. We show how $Perm_n$ is a p-projection of $f_{G_n,H_n,\mathcal{IH}}(Y)$. In any injective homomorphism, the *odd cycle* C_{k_1} attached to node u_1 in graph G_n has to get mapped

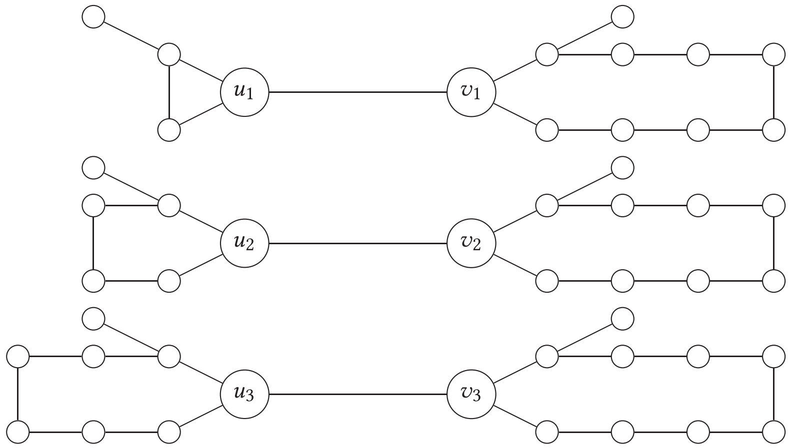


Fig. 3. G_n where $n = 3$ for $f_{G_n,H_n,\mathcal{IH}}(Y)$.

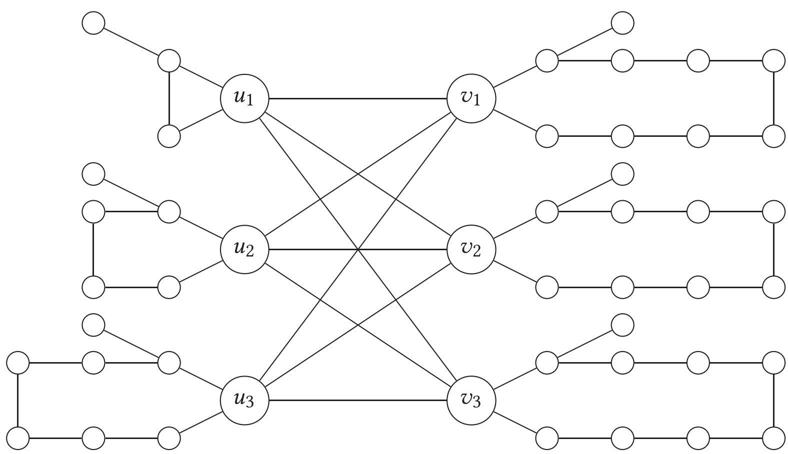


Fig. 4. H_n where $n = 3$ for $f_{G_n,H_n,\mathcal{IH}}(Y)$.

to the *odd cycle* C_{k_1} attached to node u_1' in H_n. This is because H_n has only one cycle of size k_1.[11] For any injective homomorphism to survive, u_1 in C_{k_1} of G_n has to get mapped to u_1' in C_{k_1} of H_n. By fixing the map of u_1 to u_1' from G_n to H_n, the spiked cycle allows to map C_{k_1} in G_n to C_{k_1} in H_n in only one way. Therefore, the spiked cycle $\mathcal{S}_{k_1}$ at u_1 in G_n gets exactly mapped to $\mathcal{S}_{k_1}$ at u_1' in H_n. Similarly, we can argue that any u_i in G_n gets mapped to u_i' in H_n for all $1 \leq i \leq n$ and any spiked cycle S_{k_i} at u_i in G_n gets exactly mapped to spiked cycle S_{k_i} at u_i' in H_n. It is easy to see that any v_i in G_n has to get mapped to some v_j' in H_n. Injectivity assures that no two v_i and v_j in G_n gets mapped to same v_k' in H_n. In other words, $v_1, \ldots, v_n$ in G_n gets mapped to $v_{t_1}', \ldots, v_{t_n}'$ in H_n, respectively, where the sequence $t_1, \ldots, t_n$ is any permutation of elements from $[n]$. Once we fix the mappings of $v_1, \ldots, v_n$ in G_n, the homomorphism proceeds in only way for the spiked cycles attached at $v_1, \ldots, v_n$ in G_n.

The polynomial family $f_{G_n,H_n,\mathcal{IH}}(Y)$ is not exactly the same as $Perm_n$ but has a multilinear monomial, say, α of degree $(2n + nk_{n+1} + \sum_{i=1}^{n} k_i)$ multiplied to every monomial of $Perm_n$. The

[11]Graph $\bar{H}_n$ cannot have any odd cycles as it is a bipartite graph.

variables in α are the variables associated with the spiked cycles attached to $\widehat{H_n}$ in H_n. We set all these variables to 1 to get the $Perm_n$ polynomial from $f_{G_n,H_n,\mathcal{IH}}(Y)$.

4.1.3 $f_{G_n,H_n,\mathcal{IH}}(Y)$ is in VNP. We know that $Perm_n$ is in VNP. Therefore, we have $Perm_n(\widetilde{Y}) = \sum_{Z\in\{0,1\}^{n\times n}} f_n(\widetilde{Y}, Z)$, where f_n is in VP. We know that $f_{G_n,H_n,\mathcal{IH}}(Y) = \sum_{Z\in\{0,1\}^{n\times n}} f_n'(Y,Z)$, where $f_n'(Y,Z) = \alpha.f_n(\widetilde{Y}, Z)$ and α is the multilinear monomial of degree $(2n + nk_{n+1} + \sum_{i=1}^{n} k_i)$. Clearly, f_n' is in VP, provided f_n is in VP; therefore, $f_{G_n,H_n,\mathcal{IH}}(Y)$ is in VNP.

4.2 Directed and Injective Directed Homomorphisms

In this section, we present a polynomial family that is complete for VNP, when $\mathcal{H}$ is the class of directed homomorphisms and injective directed homomorphisms. We now specify the construction and give the proof of its completeness.

4.2.1 Construction. Let G_n be a layered directed graph with four layers ℓ_1, ℓ_2, ℓ_3 and ℓ_4 each containing n nodes except layers ℓ_1 and ℓ_4, which have exactly one node each identified as the node x and the node y, respectively. We label the nodes in layer ℓ_2 as $u_1, \ldots, u_n$ and the nodes in layer ℓ_3 as $v_1, \ldots, v_n$. We add the following directed edges in G_n.

- (x, u_i) for all $1 \le i \le n$, (y, v_i) for all $1 \le i \le n$, (u_i, v_i) for all $1 \le i \le n$,
- (x, y) and (u_i, y) for all $1 \le i \le n$, (u_i, u_j) for all $i \ne j$, (v_i, v_j) for all $i < j$.

The nodes in ℓ_2 form a complete directed graph and the nodes in ℓ_3 form a tournament.

Let H_n be a layered directed graph with four layers ℓ_1, ℓ_2, ℓ_3, and ℓ_4 each containing n nodes except for layers ℓ_1 and ℓ_4. Both layers ℓ_1 and ℓ_4 has exactly one node each identified as the node x' and the node y', respectively. We label the nodes in layer ℓ_2 as $u_1', \ldots, u_n'$ and the nodes in layer ℓ_3 as $v_1', \ldots, v_n'$. We add the following directed edges in H_n.

- (x', u_i') for all $1 \le i \le m$, (y', v_i') for all $1 \le i \le n$, (u_i', v_j') for all $1 \le i, j \le n$,
- (x', y') and (u_i', y') for all $1 \le i \le n$, (u_i', u_j') for all $i \ne j$, (v_i', v_j') for all $i < j$.

The nodes in ℓ_2 form a complete directed graph and the nodes in ℓ_3 form a tournament.

For this choice of G_n, H_n, and $\mathcal{DH}$, we define the homomorphism polynomial, $f_{G_n,H_n,\mathcal{DH}}(Y)$ where $m \in \mathbb{N}$. Figure 5 shows the graphs G_n and H_n for $n = 5$.

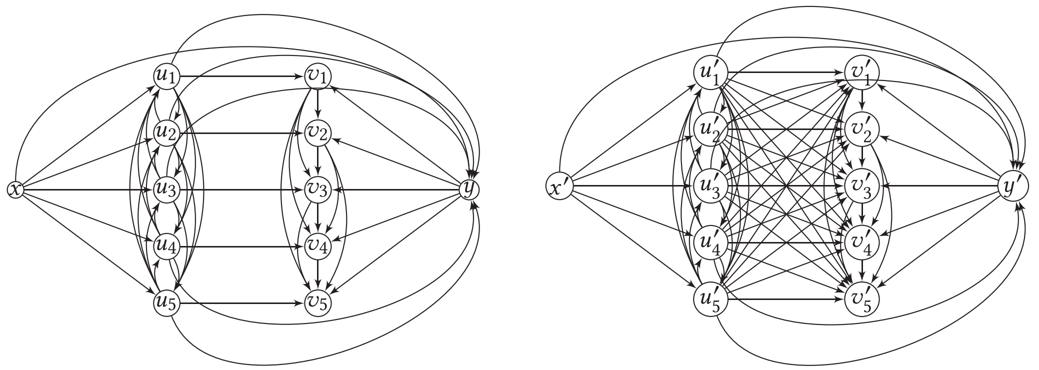
THEOREM 4.2. *The families $f_{G_n,H_n,\mathcal{DH}}(Y)$ and $f_{G_n,H_n,\mathcal{IDH}}(Y)$ are complete for the class VNP under p-projections where G_n and H_n are as described above.*

4.2.2 $f_{G_n,H_n,\mathcal{DH}}(Y)$ is VNP Hard. We show that $Perm_n$ is a p-projection of $f_{G_n,H_n,\mathcal{DH}}(Y)$.

Case I ($n = 1$): For $n = 1$, it is easy to check that in the only surviving homomorphism, x, y, u_1 and v_1 in G_n get mapped to x', y', u_1', and v_1' in H_n, respectively.

Case II ($n \ge 2$): Since the node x in G_n has a neighbourhood of $(n + 1)$ nodes that forms a clique and n nodes out of it form a complete directed graph, for any directed homomorphism to survive, x in G_n has to get mapped to x' in H_n. The neighbourhood of x, $\mathcal{N}(x)$, that is, nodes $u_1, \ldots, u_n$ and y in G_n, has to get mapped to the neighbourhood of x', $\mathcal{N}(x')$, that is, nodes $u_1', \ldots, u_n'$ and y in H_n. This $\mathcal{N}(x)$ to $\mathcal{N}(x')$ mapping has to be a bijection (this is because $\mathcal{N}(x)$ forms a clique in G_n and no node in $\mathcal{N}(x')$ has a self-loop on it.).

For any bijection to survive from $\mathcal{N}(x)$ to $\mathcal{N}(x')$ in any directed homomorphism from G_n to H_n, y in G_n must get mapped to y' in H_n. This is because if we assume that y in G_n gets mapped to some u_i' in H_n, then the remaining elements in $\mathcal{N}(x)$ must have a bijection to the remaining elements in $\mathcal{N}(x')$ but such a bijection is not possible. This is because $\mathcal{N}(x) - \{y\}$ forms a complete directed graph in G_n, whereas $\mathcal{N}(x) - \{u_i'\}$ does not form such a graph. In other words, for any



(a) G_n where $n = 5$ (b) H_n where $n = 5$

Fig. 5. Example of G_n and H_n designed for $f_{G_n,H_n,\mathcal{DH}}(Y)$.

homomorphism to survive from G_n to H_n, $u_1, \ldots, u_n, y$ in G_n gets mapped to $u'_{t_1}, \ldots, u'_{t_n}, y'$ in H_n, respectively, where the sequence $t_1, \ldots, t_n$ is any permutation of elements from $[n]$.

Once the node y in G_n gets mapped to y' in H_n, it is easy to see that the neighbourhood[12] of y, $\mathcal{N}(y)$, that is, nodes $v_1, \ldots, v_n$ in G_n, has to get mapped to the neighbourhood of y', $\mathcal{N}(y')$, that is, nodes $v'_1, \ldots, v'_n$ in H_n. This $\mathcal{N}(y)$ to $\mathcal{N}(y')$ mapping has to be a bijection (this is again because the $\mathcal{N}(y)$ forms a clique in G_n and no node in $\mathcal{N}(y')$ has a self-loop on it).

We now argue that for any bijection to survive from $\mathcal{N}(y)$ to $\mathcal{N}(y')$ in any directed homomorphism from G_n to H_n, v_i must get mapped to v'_i for all $1 \leq i \leq n$. Suppose not. In this case there exist some v_i, v_j $(i < j)$ in G_n such that v_i and v_j in G_n get mapped to $v'_{i'}$ and $v'_{j'}$ $(i' > j')$ in H_n. However, any such mapping is not a homomorphism as there is no edge $(v'_{i'}, v'_{j'})$ in H_n.

The polynomial family $f_{G_n,H_n,\mathcal{DH}}(Y)$ is not exactly the same as the $Perm_n$ but it has a multilinear monomial, say, α, of degree $3n+3\binom{n}{2}+1$ multiplied to every monomial of $Perm_n$. The variables in α are the variables associated with all the edges that are not from ℓ_2 to ℓ_3 in H_n. We set all these variables to 1 to obtain the $Perm_n$ polynomial as a p-projection of $f_{G_n,H_n,\mathcal{DH}}(Y)$.

4.2.3 $f_{G_n,H_n,\mathcal{DH}}(Y)$ is in VNP. We know that $Perm_n$ is in VNP. Therefore, we have $Perm_n(\widetilde{Y}) = \sum_{Z\in\{0,1\}^{n\times n}} f_n(\widetilde{Y}, Z)$, where f_n is in VP. We know $f_{G_n,H_n,\mathcal{DH}}(Y) = \sum_{Z\in\{0,1\}^{n\times n}} f'_n(Y, Z)$, where $f'_n(Y, Z) = \alpha.f_n(\widetilde{Y}, Z)$ and α is the multilinear monomial of degree $3n + 3\binom{n}{2} + 1$. Clearly, f'_n is in VP, therefore, $f_{G_n,H_n,\mathcal{DH}}(Y)$ is in VNP.

Observation. *It is easy to note that for any ϕ in $\mathcal{DH}$,*

- *the nodes x and y in G_n must get mapped to nodes x' and y' in H_n, respectively.*
- *nodes $u_1, \ldots, u_n$ must get mapped to $u'_{t_1}, \ldots, u'_{t_n}$, respectively, where $t_1, \ldots, t_n$ is any permutation of elements from $[n]$.*
- *For any i, v_i in G_n must get mapped to v'_i in H_n.*

This implies that any two nodes in G_n must get mapped to two distinct nodes in H_n. Therefore, $\mathcal{DH} = \mathcal{IDH}$. This observation shows that $f_{G_n,H_n,\mathcal{DH}}(Y)$ we constructed is in fact also $f_{G_n,H_n,\mathcal{IDH}}(Y)$.

[12]The neighbourhood of a node u in directed graph G, denoted by $\mathcal{N}(u)$, is the set $\{x \in V(G) | (u, x) \in E(G)\}$.

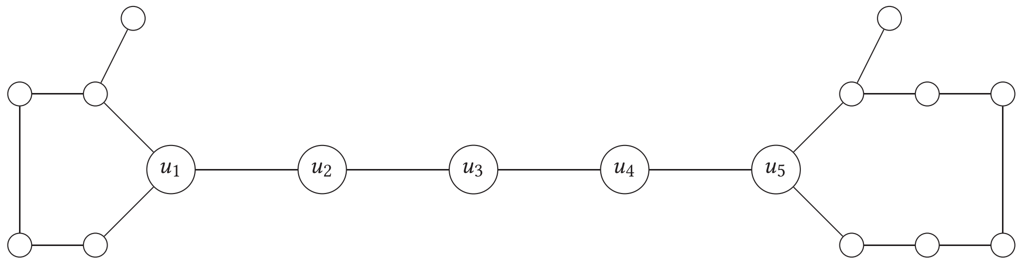


Fig. 6. G_n where $n = 4$, $k_1 = 5$, and $k_2 = 7$.

5 POLYNOMIAL FAMILIES COMPLETE FOR VBP AND VF

In this section, we present VBP and VF complete polynomial families. Before we start describing the construction we recall a well-known VBP-complete polynomial, namely $\text{IMM}_{k,n}(X)$,

$$\text{IMM}_{k,n}(X) = \sum_{i_1, i_2, \ldots, i_{n-1} \in [k]} x^{(1)}_{1,i_1} \cdot x^{(2)}_{i_1,i_2} \cdot \ldots \cdot x^{(n-1)}_{i_{n-2},i_{n-1}} \cdot x^{(n)}_{i_{n-1},1},$$

where $X = \bigcup_{\ell \in [n+1]} X^{(\ell)}$, $X^{(1)} = \{x^{(1)}_{1,j} \mid j \in [k]\}$, $X^{(n)} = \{x^{(n)}_{i,1} \mid i \in [k]\}$, and for $2 \le \ell \le n-1$, $X^{(\ell)} = \{x^{(\ell)}_{i,j} \mid i, j \in [k]\}$.

It is known that as long as $k = \Theta(\text{poly}(n))$, $\text{IMM}_{k,n}(X)$ is complete for VBP and it is complete for VF for $k = 3$ (in fact for any constant $k > 2$). (See, for instance, Reference [10].)

5.1 Injective Homomorphisms

Here we give a polynomial family that is complete for VBP, when $\mathcal{H}$ is the class of injective homomorphisms. We start with the construction of the polynomials and then present the proof of its completeness.

5.1.1 Construction. Let $\widehat{G}_n$ be a simple path on $n+1$ nodes, say, $u_1, \ldots, u_{n+1}$. We attach spiked cycles $\mathcal{S}_{k_1}$ and $\mathcal{S}_{k_2}$ to both the ends of the path, that is, at nodes u_1 and u_{n+1}, respectively. Both k_1 and k_2 are distinct odd numbers. We call this modified version of $\widehat{G}_n$ as G_n.

Let $\widehat{H}_{k,n}$ be a layered graph with $n+1$ layers labelled as $\ell_1, \ldots, \ell_{n+1}$ such that each layer has k nodes except for layers ℓ_1 and ℓ_{n+1}. The layers ℓ_1 and ℓ_{n+1} have one node each identified as the source node s and the sink node t, respectively. There are no edges within the nodes of any layer. Every node in layer i is adjacent to every other node in layer $i+1$ for all $1 \le i \le n$. We attach spiked cycles $\mathcal{S}_{k_1}$ and $\mathcal{S}_{k_2}$ to s and t in $\widehat{H}_{k,n}$, respectively. We call this modified version of $\widehat{H}_{k,n}$ as $H_{k,n}$.

For this choice of G_n, $H_{k,n}$ and $\mathcal{IH}$, we define the homomorphism polynomial, $f_{G_n, H_{k,n}, \mathcal{IH}}(Y)$ where $n \in \mathbb{N}$.

Figure 6 and Figure 7 show the graphs G_n and $H_{n,n}$ for $n = 4$, respectively.

Theorem 5.1. *The family $f_{G_n, H_{n,n}, \mathcal{IH}}(Y)$ is complete for class VBP under p-projections where G_n and $H_{n,n}$ are as described above.*

5.1.2 Hardness of $f_{G_n, H_{n,n}, \mathcal{IH}}(Y)$. We show that $\text{IMM}_{n,n}$ can be computed as a p-projection of $f_{G_n, H_{n,n}, \mathcal{IH}}(Y)$. In any injective homomorphism, the *odd cycle* C_{k_1} attached to u_1 in graph G_n has to get mapped to the *odd cycle* C_{k_1} attached to node s in $H_{n,n}$. This is because H_n has only one cycle of size k_1.[13] For any injective homomorphism to survive, the node u_1 in G_n must get mapped

[13]Graph $\bar{H}_{n,n}$ cannot have any odd cycles as it is a bipartite graph.

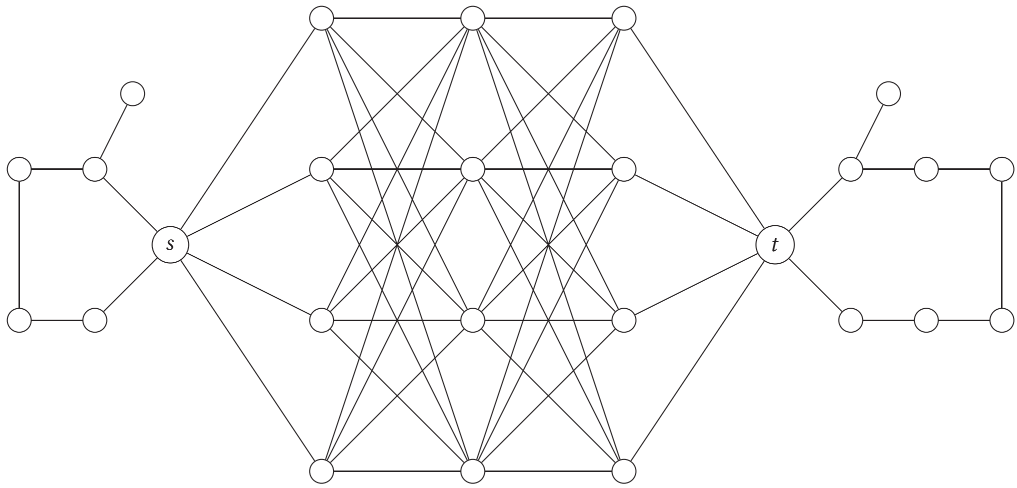


Fig. 7. H_n where $n = 4$, $k_1 = 5$, and $k_2 = 7$.

to the node s in $H_{n,n}$. Due to the spiked cycle, this mapping of C_{k_1} in G_n to C_{k_1} in $H_{n,n}$ can be done in only one way. Therefore, in any injective homomorphism mapping, the $\mathcal{S}_{k_1}$ at u_1 gets exactly mapped to $\mathcal{S}_{k_1}$ at s. Now, the nodes along the path in G_n must get mapped to the nodes across the layers in H_n. There is no possibility of folding back; this is because the node u_{n+1} in G_n has to get mapped to the node t in $H_{n,n}$; otherwise, we will not be able to map C_{k_2} at u_{n+1} in G_n to C_{k_2} at t in $H_{n,n}$.

The polynomial family we have designed is not exactly the same as the $\text{IMM}_{n,n}$ but will have a multilinear monomial, say, α, of degree $k_1 + k_2 + 2$ multiplied to every monomial of $\text{IMM}_{n,n}$. The variables in α are the variables associated with edges of the spiked cycles $\mathcal{S}_{k_1}$ and $\mathcal{S}_{k_2}$ in $H_{n,n}$. We set all these variables to 1 to obtain the $\text{IMM}_{n,n}$ polynomial from $f_{G_n,H_{n,n},\mathcal{IH}}(Y)$.

5.1.3 $f_{G_n,H_{n,n},\mathcal{IH}}(Y)$ is in VBP. We know, $f_{G_n,H_{n,n},\mathcal{IH}}(Y) = \alpha \cdot \text{IMM}_{n,n}$. Let A_n denote the algebraic branching program for $\text{IMM}_{n,n}$ with s and t as the source and sink nodes, respectively. We relabel our source node s as $\bar{s}$. We add an extra node and label it as the new source node s. We add a directed path p of length[14] $(k_1 + k_2 + 2)$ from s to $\bar{s}$. We place the variables associated with $\mathcal{S}_{k_1}$ and $\mathcal{S}_{k_2}$ in $H_{n,n}$ on the edges along the path p. We call this modified A_n as $A_n^{'}$. It is easy to note that $A_n^{'}$ computes $f_{G_n,H_{n,n},\mathcal{IH}}(Y)$.

5.2 Directed and Injective Directed Homomorphisms

In this section, we present two polynomial families complete for VBP for both directed homomorphisms ($\mathcal{DH}$) and injective directed homomorphisms ($\mathcal{IDH}$). We now specify the construction and give the proof of its completeness.

Let G_n be a simple directed path on $n + 1$ nodes, say, $u_1, \ldots, u_{n+1}$ with edges (u_i, u_{i+1}) for $1 \leq i \leq n$. Let $H_{k,n}$ be a layered graph with $n+1$ layers labelled as $\ell_1, \ldots, \ell_{n+1}$ such that each layer has k nodes except for layers ℓ_1 and ℓ_{n+1}. The layers ℓ_1 and ℓ_{n+1} have exactly one node each identified as the source node s and the sink node t, respectively. There are no edges within the nodes of any layer. There is a directed edge from every node in layer ℓ_i to every other node in layer ℓ_{i+1} for all $1 \leq i \leq n$.

[14]The length of a directed path is the number of edges along the path.

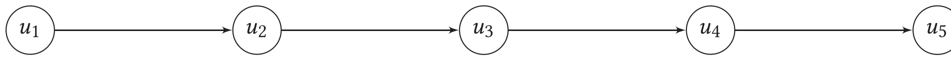


Fig. 8. G_n where $n = 4$.

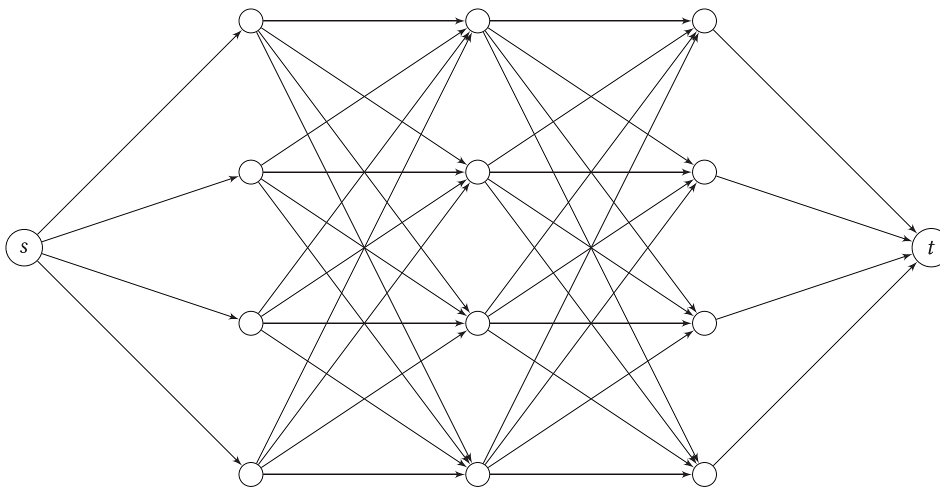


Fig. 9. H_n where $n = 4$.

Figure 8 and Figure 9 show the graphs G_n and $H_{n,n}$ for $n = 4$, respectively.

THEOREM 5.2. *The families $f_{G_n,H_{n,n},\mathcal{DH}}(Y)$ and $f_{G_n,H_{n,n},\mathcal{IDH}}(Y)$ are complete for class VBP under p-projections where G_n and $H_{n,n}$ are as described above.*

PROOF. We show that $f_{G_n,H_{n,n},\mathcal{DH}}(Y) = \mathrm{IMM}_{n,n}$. We show the bijection between the directed homomorphisms from G_n to $H_{n,n}$ to the monomials of $\mathrm{IMM}_{n,n}$. It is easy to note that for any directed homomorphism to survive from G_n to $H_{n,n}$, the node u_1 in G_n must get mapped to node s in $H_{n,n}$. It is easy to note that any directed homomorphism from G_n to $H_{n,n}$ survives if and only if the graph G_n gets exactly mapped to one of the directed paths from s to t in $H_{n,n}$. We now show that $f_{G_n,H_{n,n},\mathcal{DH}}(Y) = f_{G_n,H_{n,n}\mathcal{IDH}}(Y)$. Note that for any ϕ in $\mathcal{DH}$, the only node u in layer i in G_n must get mapped to some node v in layer i in $H_{n,n}$. Therefore, any ϕ in $\mathcal{DH}$ is also injective. Therefore, $\mathcal{DH} = \mathcal{IDH}$. This establishes $f_{G_n,H_{n,n},\mathcal{DH}}(Y) = f_{G_n,H_{n,n},\mathcal{IDH}}(Y)$. □

Remark. As $\mathrm{IMM}_{3,n}$ is complete for VF [1], we can design homomorphism polynomials complete for VF using ideas similar to those used in designing the polynomial families complete for VBP. In particular, we will use graph G_n as is in the construction and use $H_{3,n}$ to obtain polynomials $f_{G_n,H_{3,n},\mathcal{IH}}(Y)$, $f_{G_n,H_{3,n},\mathcal{DH}}(Y)$, $f_{G_n,H_{3,n},\mathcal{IDH}}(Y)$. This therefore also gives us homomorphism polynomial families complete for VF.

REFERENCES

[1] Michael Ben-Or and Richard Cleve. 1992. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.* 21, 1 (1992), 54–58.

[2] Peter Bürgisser. 2013. *Completeness and Reduction in Algebraic Complexity Theory.* Vol. 7. Springer Science & Business Media.

[3] Florent Capelli, Arnaud Durand, and Stefan Mengel. 2016. The arithmetic complexity of tensor contraction. *Theory Comput. Syst.* 58, 4 (2016), 506–527.

[4] Stephen A. Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC'71)*. 151–158.
[5] Nicolas de Rugy-Altherre. 2012. A dichotomy theorem for homomorphism polynomials. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*. Springer, 308–322.
[6] Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. 2014. Homomorphism polynomials complete for VP. In *Leibniz International Proceedings in Informatics*, Vol. 29.
[7] Christian Engels. 2016. Dichotomy theorems for homomorphism polynomials of graph classes. *J. Graph Algor. Appl.* 20, 1 (2016), 3–22.
[8] Raymond Greenlaw, H. James Hoover, and Walter Ruzzo. 1992. A compendium of problems complete for P. Citeseer.
[9] Leonid A. Levin. 1973. Universal search problems. *Probl. Inf. Transmiss.* 9, 3 (1973). [in Russian]
[10] Meena Mahajan. 2013. Algebraic complexity classes. arXiv:cs.CC/1307.3863. Retrieved from https://arxiv.org/abs/1307.3863.
[11] Meena Mahajan and Nitin Saurabh. 2018. Some complete and intermediate polynomials in algebraic complexity theory. *Theory Comput. Syst.* 62, 3 (2018), 622–652.
[12] Guillaume Malod and Natacha Portier. 2006. Characterizing valiant's algebraic complexity classes. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS'06)*. Springer, 704–716.
[13] Stefan Mengel. 2011. Characterizing arithmetic circuit classes by constraint satisfaction problems. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'11)*. Springer, 700–711.
[14] Ran Raz. 2008. Elusive functions and lower bounds for arithmetic circuits. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. ACM, 711–720.
[15] Nitin Saurabh. 2016. *Analysis of Algebraic Complexity Classes and Boolean Functions*. Ph.D. Dissertation. Institute of Mathematical sciences.
[16] L. G. Valiant. 1979. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC'79)*. 249–261.

Received August 2019; revised February 2021; accepted April 2021