Electronic Colloquium on Computational Complexity, Report No. 21 (2024)

# On the closures of monotone algebraic classes and variants of the determinant

Prasad Chaugule[1] and Nutan Limaye[2]

[1] Indian Institute of Technology, Bombay, India
[2] IT University of Copenhagen, Denmark

**Abstract.** In this paper we prove the following two results.

- We show that for any $C \in \{\mathsf{mVF}, \mathsf{mVP}, \mathsf{mVNP}\}$, $C = \overline{C}$. Here, $\mathsf{mVF}, \mathsf{mVP}$, and $\mathsf{mVNP}$ are monotone variants of $\mathsf{VF}, \mathsf{VP}$, and $\mathsf{VNP}$, respectively. For an algebraic complexity class $C$, $\overline{C}$ denotes the closure of $C$. For $\mathsf{mVBP}$ a similar result was shown in [4]. Here we extend their result by adapting their proof.
- We define polynomial families $\{\mathcal{P}(k)_n\}_{n\geq 0}$, such that $\{\mathcal{P}(0)_n\}_{n\geq 0}$ equals the Determinant polynomial. We show that $\{\mathcal{P}(k)_n\}_{n\geq 0}$ is $\mathsf{VBP}$ complete for $k = 1$ and becomes $\mathsf{VNP}$ complete when $k \geq 2$. In particular, $\{\mathcal{P}(k)_n\}$ is $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$, a polynomial obtained by summing over all signed cycle covers that avoid length $k$ cycles. We show that $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ is complete for $\mathsf{VBP}$ and $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ is complete for $\mathsf{VNP}$ for all $k \geq 2$ over any field $\mathbb{F}$.

## 1 Introduction

Valiant [19] initiated the study of the complexity of the algebraic computation. Given a polynomial, how efficiently can we compute it? In order to formalise the notion of efficiency, Valiant defined many natural models of computation. These include algebraic circuits, algebraic formulas, and algebraic branching programs.

An *algebraic circuit* is a directed acyclic graph. The nodes with in-degree zero are leaf nodes, which are labelled with variables $X = \{x_1, x_2, \ldots, x_n\}$ or field constants. The other nodes are either $+$ or $\times$ operators, which compute polynomials of their inputs. The $+$ node adds its inputs and $\times$ node multiplies its inputs. There is a designated output node which has out-degree 0. The output of the circuit is the polynomial computed by this node. An *algebraic formula* is a circuit in which the underlying DAG is a tree. The size of the circuit/formula is the number of nodes in it.

An *algebraic branching program* (ABP) $A$ is a layered DAG with two special nodes $s$ and $t$ called as the source node and the sink node, respectively. The edges are labelled with linear forms $\sum_{i=1}^{n} c_i x_i + c_0$, where $c_i \in \mathbb{F}$. For every directed path $\rho$ from $s$ to $t$, we associate a polynomial $P_\rho$ which is formed by multiplying all the labels along the edges in path $\rho$. The polynomial computed by the ABP $A$ is equal to $\sum_\rho P_\rho$ where the sum is over all $s-t$ paths in $A$ (see Definition 3.1 in [17]). The size of the algebraic branching program $A$ is the number of nodes in it. We assume that the length of every path from $s$ to $t$ is same.

ISSN 1433-8092

A polynomial family $f_n$ is called a $p$-bounded family if both the number of variables and the degree of $f_n$ are polynomially bounded in $n$. The class of $p$-bounded families computed by polynomial sized circuits is called $\mathsf{VP}$. Similarly, the class of polynomial families computed by formulas of polynomial size is called $\mathsf{VF}$. Finally, the class of polynomial families computed by ABPs of polynomial size is called $\mathsf{VBP}$.

**Closures of monotone complexity classes.** Apart from studying the complexity of the exact computation of any polynomial, we could also study the algebraic approximation of a polynomial. A polynomial $Q$ over $\mathbb{F}(\varepsilon)[X]$ is said to be an algebraic approximation of a polynomial $P$ over $\mathbb{F}[X]$ if there exist polynomials $Q_1, Q_2, \ldots$ in $\mathbb{F}[X]$ such that $Q \equiv P + \sum_{i \geq 1} \varepsilon^i Q_i$. It is possible that approximating a polynomial is computationally less expensive than computing it exactly. More formally, let $\mathcal{C}$ be any class of polynomials and $\overline{\mathcal{C}}$, the *closure of* $\mathcal{C}$, be the polynomials approximated by the polynomials in $\mathcal{C}$. One can always ask the *strict containment* question: Is $\mathcal{C} \subsetneq \overline{\mathcal{C}}$? One of the important and well-known questions is the $\mathsf{VP}$ vs. $\overline{\mathsf{VP}}$ question. Here $\overline{\mathsf{VP}}$ stands for a class of $p$-bounded families of polynomials that are approximated by polynomial sized circuits[3]. From the definition, it is clear that $\mathsf{VP} \subseteq \overline{\mathsf{VP}}$, but whether the containment is strict or not is an open question, which has a rich and long history ([3], [2], [14], [9]).

In some cases, strict containment holds. Let $\sum^{[k]} \prod \sum$ denotes the class of polynomials which can be representated as the sum of product of affine forms, where $k$ denotes the fan-in of the top-most addition gate in the circuit. Kumar [11] showed that the class of polynomials $\sum^{[3]} \prod \sum$ is strictly contained in $\overline{\sum^{[3]} \prod \sum}$. Let $\mathsf{VP}_k$ denotes the class of algebraic branching programs of width $k$. Bringmann, Ikenmeyer and Zuiddam [6] looked at the class of polynomials computed by *width*-2 *algebraic branching programs* and showed that for any constant $k$, $\mathsf{VP}_k \subseteq \overline{\mathsf{VP}_2}$. This along with a result of Allender and Wang [1] shows that $\mathsf{VP}_2 \subsetneq \overline{\mathsf{VP}_2}$.

In this work, however, we consider monotone algebraic complexity classes and for each class $\mathcal{C}$ that we consider, we show that $\mathcal{C} = \overline{\mathcal{C}}$. That is, we answer the strict containment question negatively for monotone algebraic complexity classes. Recently, Bläser, Ikenmeyer, Mahajan, Pandey and Saurabh [4] showed that $\mathsf{mVBP}$ equals its closure, where $\mathsf{mVBP}$ is a class of polynomials computable by *monotone algebraic branching program* (formal definition in Section 2). We extend their result to other monotone algebraic complexity classes. We show that $\mathsf{mVP} = \overline{\mathsf{mVP}}$, that is, anything that can be approximated by monotone circuits of polynomial size (computing polynomials of polynomial degree) can also be computed by them. We also show that $\mathsf{mVF}$, a class of polynomials computed by monotone algebraic *formulas*, is equal to its closure and that $\mathsf{mVNP}$

[3] Formally, $\overline{\mathsf{VP}}$ is defined using *topological approximations*. However, for reasonably well-behaved fields $\mathbb{F}$, the two notions of approximation are equivalent. We will focus on algebraic approximation in this note.

(formally defined in Section 3), the monotone analogue of VNP[4] equals its closure. Overall, our results imply that approximation does not add extra power to the monotone algebraic models of computation. Our proof is elementary and a simple generalisation of the proof of $\mathsf{mVBP} = \overline{\mathsf{mVBP}}$ from [4].

**Variants of the Determinant.** We recall the combinatorial definition of the determinant polynomial. (See for instance [12]).

**Definition 1.** *Let $G_n$ be the directed complete graph on $n$ vertices, where the edge label of edge $(i,j)$ is $x_{i,j}$ for $i,j \in [n]$. Let $\mathcal{C}$ denotes the set of cycle covers[5] of $G_n$. We define $\mathtt{Det_n(X)} = \sum_{C \in \mathcal{C}} s(C) \cdot m_C$, where $s(C) = (-1)^{n+t}$, where $t$ is the number of cycles in the cycle cover $C$ and $m_C$ is the monomial formed by multiplying the labels on all the edges in $C$.*

We define a variant $\mathtt{Det_n^{\neq k}(X)}$ of the determinant, which sums over all signed cycle covers that avoid length $k$ cycles. Formally,

**Definition 2.** *Let $k$ be a fixed constant. Let $G_n$ be the directed complete graph on $n$ vertices, where the edge label of edge $(i,j)$ is $x_{i,j}$ for $i,j \in [n]$. Let $\mathcal{C}^{\neq k}$ denote all those cycle covers of $G_n$ in which all the cycles have length $\neq k$. We define*

$$\mathtt{Det_n^{\neq k}(X)} = \sum_{C \in \mathcal{C}^{\neq k}} s(C) \cdot m_C$$

*where $s(C) = (-1)^{n+t}$, where $t$ is the number of cycles in the cycle cover $C$ and $m_C$ is the monomial formed by multiplying the labels on all the edges in $C$.*

We prove the following Lemma.

**Lemma 1.** $\mathtt{Det_n^{\neq 1}(X)}$ *is complete*[6] *for VBP and* $\mathtt{Det_n^{\neq k}(X)}$ *is complete for VNP for all $k \geq 2$ over any field $\mathbb{F}$.*

Using Valiants criteria ([19]), it is easy to observe that for $k \geq 0$, $\mathtt{Det_n^{\neq k}(X)}$ is in VNP. We prove VNP hardness in Section 4.

The family $\mathtt{Det_n^{\neq k}(X)}$ can be viewed as a parametrized family with $k$ as the parameter such that $\mathtt{Det_n^{\neq 1}(X)}$ is VBP-complete and $\mathtt{Det_n^{\neq k}(X)}$ for all $k \geq 2$ is VNP-complete over all fields. In previous works by Durand et.al. [8] and Chaugule et. al. [7], many polynomials were proposed which characterised VP. These were

[4] Let $f_n(x_1, x_2, \ldots, x_{k(n)})$ be a $p$-bounded polynomial family. The polynomial family $f_n$ is said to be in VNP if there exists a family $g_n \in \mathsf{VP}$ such that $f_n = \sum_{y_1, y_2 \ldots, y_{k'(n)} \in \{0,1\}} g_n(x_1, x_2, \ldots, x_{k(n)}, y_1, y_2, \ldots, y_{k'(n)})$, where $k'(n)$ is polynomially bounded in $n$.

[5] A cycle cover of a directed graph $H$ is a set of vertex disjoint cycles which are subgraphs of graph $H$ and contains all the vertices of $H$.

[6] A family $f_n(X)$ is said to be a $p$-projection of $g_n(Y)$ (denoted as $f_n(X) \leq_p g_n(Y)$) if there exists a polynomially bounded function $t : \mathbb{N} \longrightarrow \mathbb{N}$ such that $f_n(X)$ can be computed from $g_{t(n)}(Y)$ by setting its variables to one of the variables of $f_n(X)$ or to field constants. A $p$-bounded family $f_n$ is said to be hard for a complexity class $\mathcal{C}$ if for any $g_n \in \mathcal{C}$, $g_n \leq_p f_n$. A $p$-bounded family $f_n$ is complete for a class $\mathcal{C}$ if it is in $\mathcal{C}$ and is hard for $\mathcal{C}$.

defined using homomorphism polynomials. It would be interesting to come up with a parametrized polynomial family which characterizes not just VBP and VNP, but all natural complexity classes, namely VF, VBP, VP, and VNP.

## 2 Preliminaries

### 2.1 Border complexity

Analogous to algebraic complexity classes VP, VF, VBP, we define the border complexity variants of these classes. We denote the corresponding complexity classes as $\overline{\mathsf{VP}}$, $\overline{\mathsf{VF}}$, and $\overline{\mathsf{VBP}}$ respectively.

**Definition 3.** *A polynomial family $h_n \in \mathbb{F}[X]$ is said to be in class $\overline{\mathsf{VP}}$ ($\overline{\mathsf{VF}}$, or $\overline{\mathsf{VBP}}$ respectively) if and only if there exist a function $t : \mathbb{N} \longrightarrow \mathbb{N}$ and polynomials $h_{n,i} \in \mathbb{F}[X]$ such that the polynomial $g_n(x) = h_n(x) + \varepsilon h_{n,1}(x) + \varepsilon^2 h_{n,2}(x) + \ldots + \varepsilon^{t(n)} h_{n,t(n)}(x)$ can be computed by an algebraic circuit $C \in \mathbb{F}(\varepsilon)[X]$ (algebraic formula $F \in \mathbb{F}(\varepsilon)[X]$ or an algebraic branching program $A \in \mathbb{F}(\varepsilon)[X]$, respectively) of size polynomial in $n$, where $\varepsilon$ is a new indeterminate.*

### 2.2 Monotone complexity

Hereafter when we discuss results related to monotone complexity classes, we will work only with the field of real numbers or the field of rational numbers. For the sake of ease of representation, we set, $\mathbb{F} = \mathbb{R}$.

An algebraic circuit (algebraic formula and algebraic branching programs, respectively) is said to be a monotone algebraic circuit (monotone algebraic formula and monotone algebraic branching programs, respectively) if all the input constants (from $\mathbb{R}$) are positive. Note that subtractions are not allowed in the above stated models of computation.

**Definition 4.** *The class of p-bounded families computed by polynomial sized monotone circuits (monotone formula/ABPs) is called mVP (mVF, mVBP, respectively).*

### 2.3 Monotone border complexity

Analagous to our complexity classes mVP, mVF, mVBP, we now define the border complexity variants of these classes. We denote the corresponding complexity classes as $\overline{\mathsf{mVP}}$, $\overline{\mathsf{mVF}}$, $\overline{\mathsf{mVBP}}$ respectively.

Let $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$ denote the ring of Laurent polynomials that are non-negative for all sufficiently small $\varepsilon > 0$. In other words, the monotonicity condition means that for each $\alpha \in \mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$, there is a $\beta > 0$ such that for all $\varepsilon \in (0, \beta]$, $\alpha(\varepsilon) \geq 0$.

**Definition 5.** *A polynomial family $h_n \in \mathbb{F}[X]$ is said to be in class $\overline{\mathsf{mVP}}$ ($\overline{\mathsf{mVF}}$, $\overline{\mathsf{mVBP}}$) if and only if there exist a function $t : \mathbb{N} \longrightarrow \mathbb{N}$ and polynomials $h_{n,i} \in \mathbb{F}[X]$ such that the polynomial $g_n(x) = h_n(x) + \varepsilon h_{n,1}(x) + \varepsilon^2 h_{n,2}(x) + \ldots + \varepsilon^t h_{n,t(n)}(x)$ can be computed by an algebraic circuit $C$ (algebraic formula, branching program, resp.) of size polynomial in $n$ where the input are either labelled by variables or polynomials from $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$.*

The definition of $\overline{\mathsf{mVNP}}$ is slightly involved. We define it in Section 3.3.

## 3 Closures of monotone algebraic classes

The first result we prove in this section is about algebraic closure of $\mathsf{mVP}$.

**Theorem 1.** $\mathsf{mVP} = \overline{\mathsf{mVP}}$

In [4], Bläser, Ikenmeyer, Mahajan, Pandey, and Saurabh proved that "$\mathsf{mVBP} = \overline{\mathsf{mVBP}}$". Our proof has an analogous proof outline. It consists of the following three steps.

1. Let $\mathcal{C}_n$ be a monotone algebraic circuit of size $s = \text{poly}(n)$ computing the polynomial $Q = \varepsilon^0 f_0 + \sum_{i=1}^{k} \varepsilon^i f_i$. We convert the circuit $\mathcal{C}_n$ into another monotone circuit $\mathcal{C}'_n$ of size $\text{poly}(s)$, such that the circuit $\mathcal{C}'_n$ computes the polynomial $\varepsilon^t Q$ for some positive integer $t$, and no input gate in $\mathcal{C}'_n$ has a label with negative exponent of $\varepsilon$ as its input (Section 3.1).
2. We give a general procedure to convert a monotone circuit computing the polynomial $\varepsilon^t Q$ to $\varepsilon^{t-1} Q$ (This new circuit has edge labels and the construction works even if the circuit we start with has edge labels. The formal definition of a circuit with edge labels is stated in Section 3.2). Moreover, we ensure that no input gate or edge in the converted circuit has a negative exponent of $\varepsilon$ as its label. (Section 3.2)
3. We repeatedly apply step 2 and after $t$ steps construct a monotone circuit which computes the polynomial $\varepsilon^0 Q$. By finally substituting $\varepsilon$ to 0, the result follows.

Although, the overall proof idea is very similar to the proof idea from [4], the details in Step 1 and Step 2 are slightly different. Informally, we use the structural properties of the universal circuit to get Step 1, whereas we exploit the monotonicity property of the given circuit to achieve Step 2.

### 3.1 Step 1 of Theorem 1

Before going into the details of the proof this step of Theorem 1, we recall the following definitions.

**Definition 6 ([15], [18]).** *A family of circuits $\{\mathcal{U}_n\}_{n\in\mathbb{N}}$ is called a universal circuit family if for every polynomial $f_n(x_1, x_2, \ldots, x_n)$ of degree $d(n)$ which is computed by a circuit of size $s(n)$, there exists another circuit $\Psi$ which computes*

*$f_n$ such that the underlying Directed Acyclic Graph (DAG) of $\Psi$ is the same as that of $\mathcal{U}_m$ for some $m \in \text{poly}(n, s, d)$. We assume that both $s(n)$ and $d(n)$ are polynomially bounded in $n$.*

*A universal circuit $C_n$ is said to be in a normal form if it satisfies the following properties.*

- *The circuit $C_n$ is a semi-bounded circuit, i.e., the indegree of $\times$ gate is 2, whereas the indegree of $+$ gates is unbounded.*
- *The circuit $C_n$ is a multiplicatively disjoint circuit.*
- *The circuit $C_n$ is a layered circuit where layers are alternately labelled with $+$ and $\times$ gates. We assume that the root node is a $\times$ gate. Also, the distance between the root node to every leaf node of circuit $C_n$ is the same.*
- *The degree of the circuit $C_n$ is $n$. The depth of the circuit $C_n = 2c\lceil \log n \rceil$, for some constant $c$.*
- *The number of variables $v(n)$ and the size $s(n)$ of the circuit $C_n$ are both polynomially bounded in $n$.*

*It is known that a universal circuit can be assumed to be in a normal form.*

We recall the definition of a parse tree of an algebraic circuit.

**Definition 7 ([13]).** *The set of parse trees of a circuit $C$, $\mathcal{T}(C)$ is defined inductively on the size of circuit $C$*

- *If the size of $C$ is 1, then the circuit $C$ is its own parse tree.*
- *If the circuit size is greater than 1, then the output gate is either a multiplication gate or an addition gate*
  1. *Let the output gate $t$ be an addition gate. Then the parse trees of circuit $C$ are formed by taking a parse tree of any one of its children, say $t'$ along with the edge $(t', t)$.*
  2. *Let the output gate $t$ be a multiplication gate. Let $t_1$ and $t_2$ be the children gates of $t$. Let $C_{t_1}$ and $C_{t_2}$ be the subcircuits rooted at gates $t_1$ and $t_2$ in circuit $C$. The parse trees of $C$ are formed by taking a node disjoint copy of a parse tree of subcircuit $C_{t_1}$ and a parse tree of subcircuit $C_{t_2}$ along with the edges $(t_1, t)$ and $(t_2, t)$.*

*Remark 1.* Given an algebraic circuit $C$ computing a polynomial $f$. Let $\mathcal{T}(C)$ denote the set of all parse trees of circuit $C$. Let $mon(T)$ be the monomial associated with $T \in \mathcal{T}(C)$, where $mon(T)$ is equal to the product of the labels of the leaves of $T$. It is known that $f$ is equal to $\sum_{T \in \mathcal{T}(C))} mon(T)$ (see [13]).

*Remark 2.* Note that the shape of every parse tree $T$ of the universal circuit (in normal form) $C_n$ is the same. Moreover, the number of leaf nodes in any parse tree $T$ of $C_n$ is equal to $2^{c\lceil \log n \rceil} = n^c$.

The following Lemma gives the details for the first step.

**Lemma 2.** *Consider a family of monotone algebraic circuit $\mathcal{C}_n$ of size $s = \text{poly}(n)$ computing a polynomial $Q = \varepsilon^0 f_0 + \sum_{i=1}^{k} \varepsilon^i f_i$ (that is, circuit $\mathcal{C}_n$ computes the polynomial $f_0$ in the border sense), then there exists another monotone circuit $\mathcal{C}'_n$ (in normal form) of size $\text{poly}(s)$ such that the circuit $\mathcal{C}'_n$ computes the polynomial $\varepsilon^t Q$, for some positive integer $t$. Moreover, no input gate in circuit $\mathcal{C}'_n$ has a label with a negative exponent of $\varepsilon$ as its input. And the underlying DAG structure of $\mathcal{C}'_n$ is the same as that of $\mathcal{C}_n$.*

*Proof.* Consider a circuit $\mathcal{C}_n$ which computes the polynomial $Q$ and say it has size $s$. We know that there exists a universal circuit in normal form $\mathcal{U}_m$, where $m = \text{poly}(s)$ such that the circuit $\mathcal{U}_m$ also computes $Q$. Moreover, it is known that the monotonicity of circuit $\mathcal{C}_n$ can be preserved in circuit $\mathcal{U}_m$. Let $j$ be the largest negative exponent in any of the input gates of $\mathcal{C}_n$. We multiply the label of every input gate of circuit $\mathcal{U}_m$ by $\varepsilon^j$. We call this new circuit $\mathcal{C}'_n$. Since, the circuit $\mathcal{C}_n$ is in the normal form, the shape of every parse tree of circuit $\mathcal{C}_n$ is the same and the number of leaf nodes in any parse tree is equal to $n^c$ (see Remark 1 and Remark 2). Therefore, the polynomial computed by circuit $\mathcal{C}'_n$ is equal to a scaled version of polynomial computed by circuit $\mathcal{C}_n$, that is, the polynomial computed by $\mathcal{C}'_n$ is equal to $\varepsilon^t Q$, where $t = j \times n^c$. We did not change the underlying graph structure of $\mathcal{C}_n$ to obtain $\mathcal{C}'_n$. □

### 3.2 Step 2 of Theorem 1

We now present the details regarding the second step of our main proof.

Here we will use algebraic circuits with edge labels. A circuit $C$ with edge labels over a field $\mathbb{F}$ and variable set $X = \{x_1, x_2, \ldots, x_n\}$ is exactly similar to the algebraic circuit we have been considering, except for the edge function $w : E \longrightarrow X \cup \mathbb{F}$. Here $E$ denotes the edge set of $C$. The polynomial computed at any input gate $u$ is equal to the label of $u$. Let $P_u$ denote the polynomial computed at the node $u$ in the circuit $C$. The polynomial computed at any computation gate $u$ with operation $\texttt{op}$, with $u_\ell$ and $u_r$ as its children nodes is equal to the $(w((u, u_\ell)) \times P_{u_\ell}) \texttt{op} (w((u, u_r)) \times P_{u_r})$. The polynomial computed by the circuit $C$ is the polynomial computed by the output gate of circuit $C$. The size of the algebraic circuit $C$ is the number of nodes in it.

**Definition 8.** *A node $u$ in any monotone circuit over $\mathbb{R}_+[\varepsilon^{-1}, \varepsilon]$ is called a good node, if the polynomial $f_u$ computed at node $u$ is divisible by $\varepsilon$.*

**Lemma 3.** *Consider any monotone circuit family $\mathcal{D}_n$ of size $s$ with edge label function $w : E \longrightarrow \{\varepsilon^i | i \in \mathbb{Z}_{\geq 0}\}$, where $E$ is the set of edges of $\mathcal{D}_n$. Suppose it computes a polynomial $\varepsilon^b Q$ for some $b \geq 1$ where $Q = f_0 + \sum_{i=1}^{b} \varepsilon^i f_i$ and the circuit $\mathcal{D}_n$ satisfies the following properties:*
*- No input gate has a label with a negative exponent of $\varepsilon$.*
*- The underlying graph structure of $\mathcal{D}_n$ is the same as the graph structure of a universal circuit as in Definition 6.*

*Then there exists a circuit $\mathcal{D}'_n$ of size $s$ such that $\mathcal{D}'_n$ computes $\varepsilon^{b-1} Q$. Moreover, no input gate (or an edge label) in $\mathcal{D}'_n$ is labelled with a negative*

*exponent of $\varepsilon$. Also, the underlying graph structure of the circuit $\mathcal{D}'_n$ is same as that of $\mathcal{D}_n$, upto the relabelling of the input gates and edge labels of the circuit $\mathcal{D}_n$.*

Let $\mathcal{G}$ denote the set of all good nodes of circuit $\mathcal{D}_n$. In order to prove Lemma 3, we need to describe the circuit $\mathcal{D}'_n$. The circuit $\mathcal{D}'_n$ is exactly similar to the circuit $\mathcal{D}_n$, with labels of some input nodes $\mathcal{L}' \subseteq \mathcal{G}$ scaled by $1/\epsilon$, and an updated edge function $w'$. Moreover, the new function $w'$ also satisfies the property that no edge has a label with a negative exponent of $\varepsilon$.

Instead of $\mathcal{L}'$, we construct a larger set $\mathcal{S}$ such that $\mathcal{L}' \subseteq \mathcal{S} \subseteq \mathcal{G}$ and the root node $r \in \mathcal{S}$. Let $f_u$ denote the polynomial computed at node $u$ in circuit $\mathcal{D}_n$. Let $\widehat{f_u}$ denote the polynomial computed at node $u$ in circuit $\mathcal{D}'_n$. It suffices to prove the following lemma.

**Lemma 4.** *In the circuit $\mathcal{D}'_n$, for every node $u \in \mathcal{S}$, $\widehat{f_u} = \frac{1}{\varepsilon} f_u$.*

Since, the root node $r \in \mathcal{S}$, the proof of Lemma 3 immediately follows (see Section 3.2 for proof details).

**Identifying the set $\mathcal{S}$ and the edge label function $\widehat{w}$** The main idea is to mark some of the good nodes of circuit $\mathcal{D}_n$ which will form the set $\mathcal{S}$. First of all, we will mark the root node of circuit $\mathcal{D}_n$. The main goal is to scale the polynomial computed at the marked root node by $1/\epsilon$. We will propagate this effect from the root node layer to the layer immediately after it and so on till we reach to the layer of the leaf nodes. In this process, we may also change the labels of the edges thereby changing the old $w$ function to the new updated $w'$ function. Upon reaching the last layer, we scale all the marked leaf nodes of $\mathcal{D}_n$ by $1/\epsilon$ (by relabelling). We now discuss the procedure in detail.

Let us number the layers of $\mathcal{D}_n$ from 1 to $m$ such that the layer containing the root node is numbered as 1 and so on till the layer containing the leaf nodes is numbered $m$.

**Marking the root node**: We mark the root node of circuit $\mathcal{D}_n$. Note that by our assumption of circuit $\mathcal{D}_n$, the root node is always a good node.

**Marking the nodes at layer $i+1$** : Given the marking of nodes upto layer numbered $i$, we give a procedure to mark the nodes in layer numbered $i+1$. We break this case into two parts depending on whether $i+1$ is even or odd.

**Case 1: $i+1$ is odd and $i+1 \leq m$** : We know that the layer numbered $i$ consists of the $+$ gates.. Let $u_1, u_2, \ldots, u_t$ be the marked nodes of layer $i$. Inductively, we know that $u_1, u_2, \ldots, u_t$ are all good nodes. For each $j \in [t]$, let $u_{j,1}, u_{j,2}, \ldots, u_{j,f(j)}$ be children of node $u_j$. Note that, for all $j \in [t]$, the nodes $u_{j,1}, u_{j,2}, \ldots, u_{j,f(j)}$ are in layer numbered $i+1$. For each $k \in [f(j)]$, the condition of the monotonicity of $\mathcal{D}_n$ (along with the property that $u_j$ is a good node and is an addition gate) guarantees that

1a. either $u_{j,k}$ is a good node,
1b. or the edge $(u_{j,k}, u_j)$ is labelled by $\varepsilon^\beta$ for some $\beta \geq 1$.

We now describe a procedure which essentially scales the polynomials computed via $u_{j,1}$, $u_{j,2}$, $\ldots$, $u_{j,f(j)}$ by $1/\varepsilon$, which in turn implies that the polynomial computed at $u_j$ is also scaled by $1/\varepsilon$. This scaling is done either immediately by reducing the exponent of $\varepsilon$ along the edge between the child node and the $+$ gate or is postponed to the next layer by marking the child node. We now state it in Algorithm 1 below.

```
for j = 1 to t do
    for k = 1 upto f(j) do
        if the node u_{j,k} is already marked then
            we do nothing;
        end
        else if Case 1a holds then
            we mark the node u_{j,k};
        end
        else if Case 1b holds then
            we relabel the edge (u_{j,k}, u_j) initially labelled by ε^β to ε^{β-1};
        end
    end
end
```

**Algorithm 1:** Procedure to mark nodes on layer $i+1$ when $i+1$ is odd.

**Case 2: $i+1$ is even and $i+1 \leq m$** : We know that the layer numbered $i$ consists of $\times$ gates. Let $u_1, u_2, \ldots, u_t$ be the marked nodes of layer $i$. Inductively, we know that $u_1, u_2, \ldots, u_t$ are all good nodes. For each $j \in [t]$, let $u_{j,\ell}$ and $u_{j,r}$ be the left child and the right child of node $u_j$, respectively. Note that, for all $j \in [t]$, the nodes $u_{j,\ell}$ and $u_{j,r}$ are in layer numbered $i+1$. Since the circuit $\mathcal{D}_n$ is multiplicatively disjoint, for each $j \in [t]$, nodes $u_{j,\ell}$ and $u_{j,r}$ are always distinct. The property of monotonicity of $\mathcal{D}_n$ (along with the property that $u_j$ is a good node and is a multiplication gate) guarantees that one of the following two cases holds.

2a. If both $u_{j,\ell}$ and $u_{j,r}$ are not good nodes then there exist at least one edge from $\{(u_{j,\ell}, u_j), (u_{j,r}, u_j)\}$ which is labelled by $\varepsilon^\gamma$ for some $\gamma \geq 1$. Let us call that edge $e'$.

2b. Either $u_{j,\ell}$ or $u_{j,r}$ is a good node. (If both are good we arbitrarily fix one and call it as $z$)

The main idea of the following procedure is to scale either the polynomial fed via $u_{j,\ell}$ or the polynomial fed via $u_{j,r}$ by $1/\varepsilon$, which in turn scales the polynomial computed at $u_j$ by $1/\varepsilon$. This scaling is done either immediately by reducing/increasing the exponent of $\varepsilon$ appropriately along the edge between the child node and the $\times$ gate and/or is postponed to the next layer by marking the child node. We now state the procedure in Algorithm 2 below.

Set $\mathcal{E} = \phi$
**for** $j = 1$ *to* $t$ **do**
  **if** *exactly one of the nodes from* $u_{j,\ell}$ *and* $u_{j,r}$ *is marked* **then**
    we do nothing
  **end**
  **else if** *both the nodes* $u_{j,\ell}$ *and* $u_{j,r}$ *are marked* **then**
    we arbitrarily pick one of the edges from $\{(u_{j,\ell}, u_j), (u_{j,r}, u_j)\}$. Let us call that edge $e$. we increase the exponent of $\varepsilon$ on edge $e$ by 1.
  **end**
  **else if** *Case 2a holds* **then**
    we reduce the exponent of $\varepsilon$ on edge $e'$ by 1, that is, we relabel the edge $e'$ by $\varepsilon^{\gamma-1}$.
  **end**
  **else if** *Case 2b holds* **then**
    we mark the node $z$. Let $\mathcal{K} \subseteq \mathcal{E}$ such that each node $a \in \mathcal{K}$ is a parent of $z$.
    **for** *each* $a \in \mathcal{K}$ **do**
      the exponent of the $\varepsilon$ on edge $(z, a)$ is increased by 1.
    **end**
  **end**
  Update $\mathcal{E} = \mathcal{E} \cup \{u_j\}$.
**end**

**Algorithm 2:** Procedure to mark nodes on layer $i+1$ when $i+1$ is even.

We know that $\mathcal{S}$ consists of all the marked nodes of $\mathcal{D}_n$. Let $\ell_1, \ell_2, \ldots, \ell_{t'}$ be the marked leaf nodes of $\mathcal{D}_n$. By the construction, we know that these nodes are good nodes. Therefore, their labels have the exponent of $\varepsilon$ at least 1. We reduce the exponent of $\varepsilon$ in each of these marked leaves by 1. We call the new circuit $\mathcal{D}'_n$ where $w'$ denotes the edge label function of $\mathcal{D}'_n$.

**Proof of Lemma 4** We prove by using induction on the layer numbers of the nodes in $\mathcal{S}$ in the circuit $\mathcal{D}'_n$.
**Base Case:** Consider the layer $m$ of $\mathcal{D}'_n$. This layer consists of the leaf nodes of circuit $\mathcal{D}'_n$. By our construction, it is easy to note that the base case holds.
**Inductive case:** We assume that the inductive hypothesis holds for layer numbered $i+1$ and show that it holds for layer $i$. We break this case into two parts depending on whether $i$ is even or odd.
***i* is even:** We know that the layer numbered $i$ consists of the addition gates. Let $u_1, u_2, \ldots, u_t$ be the marked nodes of layer $i$. For each $j \in [t]$, let $u_{j,1}, u_{j,2}, \ldots, u_{j,f(j)}$ be children of node $u_j$. For each $j \in [t]$ we know that $f_{u_j} = \sum_{k=1}^{f(j)} w(u_{j,k}, u_j) \times f_{u_{j,k}}$ and therefore, $\widehat{f}_{u_j} = \sum_{k=1}^{f(j)} w'(u_{j,k}, u_j) \times \widehat{f}_{u_{j,k}}$. By our construction, for all $k \in [f(j)]$, either $u_{j,k}$ is a marked node or the edge label function $w'$ updates the weight on edge $(u_{j,k}, u_j)$ as follows : $w'(u_{j,k}, u_j) = \frac{1}{\varepsilon} w(u_{j,k}, u_j)$. Let $\mathcal{M} \subseteq [f(j)]$ is such that for each $a \in \mathcal{M}$, $u_{j,a}$ is

a marked node. Inductively, we know that $\widehat{f}_{u_{j,a}} = \frac{1}{\varepsilon} f_{u_{j,a}}$. Therefore,

$$\widehat{f}_{u_j} = \sum_{k=1}^{f(j)} w'(u_{j,k}, u_j) \times \widehat{f}_{u_{j,k}} \tag{1}$$

$$= \sum_{a \in \mathcal{M}} w'(u_{j,a}, u_j) \times \widehat{f}_{u_{j,a}} + \sum_{b \in \overline{\mathcal{M}}} w'(u_{j,b}, u_j) \times \widehat{f}_{u_{j,b}} \tag{2}$$

$$= \sum_{a \in \mathcal{M}} w(u_{j,a}, u_a) \times \frac{1}{\varepsilon} f_{u_{j,a}} + \sum_{b \in \overline{\mathcal{M}}} \frac{1}{\varepsilon} w(u_{j,b}, u_j) \times f_{u_{j,b}} = \frac{1}{\varepsilon} f_{u_j} \tag{3}$$

***i* is odd** : We know that the layer numbered $i$ consists of the multiplication gates. Let $u_1, u_2, \ldots, u_t$ be the marked nodes of layer $i$. For each $j \in [t]$, let $u_{j,\ell}$ and $u_{j,r}$ be the left child and the right child of node $u_j$ repectively. For each $j \in [t]$ we know that $f_{u_j} = w(u_{j,\ell}, u_j) \times f_{u_{j,\ell}} \times w(u_{j,r}, u_j) \times f_{u_{j,r}}$ and therefore, $\widehat{f}_{u_j} = w'(u_{j,\ell}, u_j) \times \widehat{f}_{u_{j,\ell}} \times w'(u_{j,r}, u_j) \times \widehat{f}_{u_{j,r}}$.

By the given procedure, the set of nodes $u_j$ can always be partitioned into three sets, say, $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. Let $\mathcal{M}_1$ be the set such that for all $a \in \mathcal{M}_1$, exactly one of the child nodes from $u_{a,\ell}$ and $u_{a,r}$ is a marked node. Let $\mathcal{M}_2$ be the set such that for all $b \in \mathcal{M}_2$, both the child nodes $u_{b,\ell}$ and $u_{b,r}$ are not marked and either $w'(u_{b,\ell}, u_b)$ is set to be equal to $\frac{1}{\varepsilon} w(u_{b,\ell}, u_b)$ or $w'(u_{b,r}, u_b)$ is set to be equal to $\frac{1}{\varepsilon} w(u_{b,r}, u_b)$. Let $\mathcal{M}_3$ be the set such that for all $c \in \mathcal{M}_3$, both the child nodes are marked and either $w'(u_{c,\ell}, u_c)$ is set to be equal to $(\varepsilon \times w(u_{c,\ell}, u_c))$ or $w'(u_{c,r}, u_c)$ is set to be equal to $(\varepsilon \times w(u_{b,r}, u_b))$.

For any $a \in \mathcal{M}_1$, we either have

$$\widehat{f}_{u_a} = w'(u_{a,\ell}, u_a) \times \widehat{f}_{u_{a,\ell}} \times w'(u_{a,r}, u_a) \times \widehat{f}_{u_{a,r}} \tag{4}$$

$$= w(u_{a,\ell}, u_a) \times \frac{1}{\varepsilon} f_{u_{a,\ell}} \times w(u_{a,r}, u_a) \times f_{u_{a,r}} = \frac{1}{\varepsilon} f_{u_a} \tag{5}$$

or

$$\widehat{f}_{u_a} = w'(u_{a,\ell}, u_a) \times \widehat{f}_{u_{a,\ell}} \times w'(u_{a,r}, u_a) \times \widehat{f}_{u_{a,r}} \tag{6}$$

$$= w(u_{a,\ell}, u_a) \times f_{u_{a,\ell}} \times w(u_{a,r}, u_a) \times \frac{1}{\varepsilon} f_{u_{a,r}} = \frac{1}{\varepsilon} f_{u_a} \tag{7}$$

For any $b \in \mathcal{M}_2$, we either have

$$\widehat{f}_{u_b} = w'(u_{b,\ell}, u_b) \times \widehat{f}_{u_{b,\ell}} \times w'(u_{b,r}, u_b) \times \widehat{f}_{u_{b,r}} \tag{8}$$

$$= \frac{1}{\varepsilon} w(u_{b,\ell}, u_b) \times f_{u_{b,\ell}} \times w(u_{b,r}, u_b) \times f_{u_{b,r}} = \frac{1}{\varepsilon} f_{u_b} \tag{9}$$

or

$$\widehat{f}_{u_b} = w'(u_{b,\ell}, u_b) \times \widehat{f}_{u_{b,\ell}} \times w'(u_{b,r}, u_b) \times \widehat{f}_{u_{b,r}} \tag{10}$$

$$= w(u_{b,\ell}, u_b) \times f_{u_{b,\ell}} \times \frac{1}{\varepsilon} w(u_{b,r}, u_b) \times f_{u_{b,r}} = \frac{1}{\varepsilon} f_{u_b} \tag{11}$$

For any $c \in \mathcal{M}_3$, we either have

$$\widehat{f}_{u_c} = w'(u_{c,\ell}, u_c) \times \widehat{f}_{u_{c,\ell}} \times w'(u_{c,r}, u_c) \times \widehat{f}_{u_{c,r}} \tag{12}$$

$$= w(u_{c,\ell}, u_c) \times \frac{1}{\varepsilon} f_{u_{c,\ell}} \times (\varepsilon \times w(u_{c,r}, u_c)) \times \frac{1}{\varepsilon} f_{u_{c,r}} = \frac{1}{\varepsilon} f_{u_c} \tag{13}$$

or

$$\widehat{f}_{u_c} = w'(u_{c,\ell}, u_c) \times \widehat{f}_{u_{c,\ell}} \times w'(u_{c,r}, u_c) \times \widehat{f}_{u_{c,r}} \tag{14}$$

$$= (\varepsilon \times w(u_{c,\ell}, u_c)) \times \frac{1}{\varepsilon} f_{u_{c,\ell}} \times w(u_{c,r}, u_c) \times \frac{1}{\varepsilon} f_{u_{c,r}} = \frac{1}{\varepsilon} f_{u_c} \tag{15}$$

This finishes the proof.

By repeatedly applying the step 2, we obtain a circuit $\mathcal{C}'_n$ which computes the polynomial $\varepsilon^0 Q$. This finishes the proof of Theorem 1.

### 3.3 mVNP and mVF

We state the definition of the monotone variant of VNP.

**Definition 9.** *Let $f_n(x_1, x_2, \ldots, x_{k(n)})$ be a p-bounded polynomial family. We say that $f_n$ is in mVNP if there exists a family $g_n \in$ mVP such that $f_n = \sum_{y_1, y_2 \ldots, y_{k'(n)} \in \{0,1\}} g_n(x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_{k'(n)})$, where $k'(n)$ is polynomially bounded in $n$.*

We now recall the definition of $\overline{\mathsf{VNP}}$ from [10].

**Definition 10 ([10]).** *Let $f_n(x_1, x_2, \ldots, x_{k(n)})$ be a p-bounded polynomial family. We say that $f_n$ is in $\overline{\mathsf{VNP}}$ if there exists a p-bounded family $\hat{f}_n \in$ VNP over the field $\mathbb{F}(\varepsilon)$ and a function $t : \mathbb{N} \longrightarrow \mathbb{N}$ such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \ldots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \ldots, f_{n,t(n)}$ defined over $\mathbb{F}$.*

We can now define the class $\overline{\mathsf{mVNP}}$.

**Definition 11.** *Let $f_n(x_1, x_2, \ldots, x_{k(n)})$ be a p-bounded polynomial family. We say that $f_n$ is in $\overline{\mathsf{mVNP}}$ if there exist a p-bounded family $\hat{f}_n \in$ mVNP over $\mathbb{R}_+[\varepsilon, \varepsilon^{-1}]$ and a function $t : \mathbb{N} \longrightarrow \mathbb{N}$ such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \ldots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \ldots, f_{n,t(n)}$ defined over $\mathbb{R}$.*

**Lemma 5.** *mVNP $= \overline{\mathsf{mVNP}}$*

*Proof.* Let $f_n(x_1, x_2, \ldots, x_{k(n)}) \in \overline{\mathsf{mVNP}}$. By the definition of $\overline{\mathsf{mVNP}}$, we know that there exists a $p$-family $\hat{f}_n \in \mathsf{mVNP}$ (over the field $\mathbb{F}(\varepsilon)$) such that $\hat{f}_n(x) = f_n(x) + \varepsilon f_{n,1}(x) + \varepsilon f_{n,2}(x) + \ldots + \varepsilon^{t(n)} f_{n,t(n)}(x)$ for some $f_{n,1}, f_{n,2}, \ldots, f_{n,t(n)} \in \mathbb{R}[X]$. We know that $\hat{f}_n = \sum_{y_1, y_2 \ldots, y_{k'(n)} \in \{0,1\}} g_n(X, Y)$, where $k'(n)$ is polynomially bounded in $n$ and $g_n(X, Y) \in \mathsf{mVP}$ (over the field $\mathbb{F}(\varepsilon)$). Let $g_n(X, a_1, a_2, \ldots, a_{k'(n)})$ denote the evaluation of the polynomial $g_n(X, Y)$ at $y_1 = a_1$, $y_2 = a_2$, $\ldots$, $y_k = a_{k'(n)}$. Since the polynomial $\hat{f}_n$ is monotone and it

converges, each of the summand in $\hat{f}_n = \sum_{y_1,y_2\ldots,y_{k'(n)}\in\{0,1\}} g_n(X,Y)$ must also necessarily converge. That is, for any boolean setting of variables $y_1 = a_2, y_2 = a_2, \ldots, y_{k'(n)} = a_{k'(n)}$, the polynomial $g_n(X, a_1, a_2, \ldots, a_{k'(n)})$ must converge. Therefore, $g_n(X,Y)$ must also converge. By Theorem 1, there exists a circuit $C_{g_n}$ which computes $g_n(X,Y)$ such that there are no negative exponents of $\varepsilon$ in any of its labels. Let $g_n(X,Y)_{\varepsilon=0}$ denote the polynomial obtained after substituting $\varepsilon = 0$ in $g_n(X,Y)$. We know that $f_n(x) = \sum_{y_1,y_2\ldots,y_{k'(n)}\in\{0,1\}} g_n(X,Y)_{\varepsilon=0}$. By the definition of $\mathsf{mVNP}$, the result follows. □

We now state the final lemma of this section. We use the proof of $\mathsf{mVBP} = \overline{\mathsf{mVBP}}$ from [4] to prove Lemma 6.

**Lemma 6.** $mVF = \overline{mVF}$

*Proof.* Consider a monotone formula $F_n$ of size $s = \text{poly}(n)$ which computes a polynomial $Q = \varepsilon^0 f_0 + \sum_{i=1}^{k} \varepsilon^i f_i$. We convert $F_n$ into an algebraic branching program $B_{F_n}$ of size poly$(s)$ such that the polynomial computed by $B_{F_n}$ is $Q$. It is easy to observe that the monotonicity of $F_n$ is preserved in $B_{F_n}$ and the underlying graph of $B_{F_n}$ is a series-parallel graph (see [5] for details about series-parallel graphs). Since, "$\mathsf{mVBP} = \overline{\mathsf{mVBP}}$", there exists another monotone algebraic branching program $\widehat{B_{F_n}}$ of size poly$(s)$ which exactly computes $f_0$. It is easy to observe (from the proof of "$\mathsf{mVBP} = \overline{\mathsf{mVBP}}$" in [4]) that the underlying graphs of both the algebraic branching programs $B_{F_n}$ and $\widehat{B_{F_n}}$ are isomorphic to each other (where the source and sink of $B_{F_n}$ maps with the source and sink of $\widehat{B_{F_n}}$ respectively) and therefore, the underlying graph of $\widehat{B_{F_n}}$ is also a series-parallel graph of size poly$(s)$. It is easy to note that an algebraic branching program $\widehat{B_{F_n}}$ of size poly$(s)$ computing polynomial $f_0$ with an underlying series-parallel graph can always be associated with a series-parallel tree. We can inductively construct the monotone algebraic formula $\widehat{F}_n$ computing $f_0$ (of size $s$) back from the series-parallel tree .

## 4 Complexity of $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$

It is not very hard to see that $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ is $\mathsf{VBP}$ complete for $k = 1$. The proof idea is similar to the proof which shows that $\mathtt{Det}_{\mathtt{n}}(\mathtt{X})$ is $\mathsf{VBP}$ complete.

**Lemma 7.** $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ *is complete for VBP (under p-projections) for all fields* $\mathbb{F}$.

*Proof.* It is easy to note that $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ is a projection of $\mathtt{Det}_{\mathtt{n}}(\mathtt{X})$ where for all $1 \leq i \leq n$, $x_{i,i}$ is set to 0. Therefore, it immediately follows that $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ is in $\mathsf{VBP}$. We show that $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ is hard for $\mathsf{VBP}$ under $p$-projections. Let $f_m$ be a polynomial computed by a layered algebraic branching program $A_m$ of size poly$(m)$. It is sufficient to show that $f_m$ can be computed as a projection of $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ where $n$ is poly$(m)$. Let $s$ and $t$ be the source and sink vertices of $A_m$, respectively. We describe a graph gadget $\mathcal{G}$. Let $V(\mathcal{G}) = \{v_1, v_2, v_3\}$ and $E(\mathcal{G}) = \{(v_1, v_2), (v_2, v_3), (v_3, v_2), (v_3, v_1)\}$. Every edge in $\mathcal{G}$ has weight 1. We

say that gadget $\mathcal{G}$ is identified with any vertex $u$ of $A_m$ iff vertex $v_1$ of $\mathcal{G}$ is merged/identified with $u$. We will convert $A_m$ to another graph $G_n$ such that under our projection, $\mathtt{Det}_{\mathtt{n}}^{\neq 1}(\mathtt{X})$ is $f_m$ and $n = \text{poly}(m)$. We will now give the steps to convert $A_m$ to $G_n$.

1. We add an extra vertex $\alpha$ and add a directed edge from $t$ to $\alpha$ and another directed edge from $\alpha$ to $s$.
2. We identify every vertex of $A_m$ (except the vertex $\alpha$) with distinct copies of $\mathcal{G}$.

Note that $n = \text{poly}\,(m)$ and the graph $G_n$ does not have any self-loops. Therefore, it is sufficient to show that the determinant of $G_n$ is $f_m$. Since $\alpha$ does not have a self-loop on it, the only way to cover $\alpha$ in any cycle cover of $G_n$ is by some cycle $\mathcal{C}$ which uses the edges $(\alpha, s)$ and then a directed path $\mathcal{P}_{\mathcal{C}}$ from $s$ to $t$ followed by the edge, $(t, \alpha)$. It is easy to see that all the vertices of $G_n$ which are not covered by cycle $\mathcal{C}$ are all gadget vertices of some copy of $\mathcal{G}$. It is easy to observe that there exists a unique way to cover all the remaining vertices of $G_n$ which are not used in $\mathcal{C}$. Note that the total number of cycles in any cycle cover of $G_n$ is the same; therefore, the sign is the same for all the cycle covers. This finishes the proof □

**Lemma 8.** $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ *is complete (under p-projections) for* VNP *for all* $k \geq 2$ *for all fields* $\mathbb{F}$.

**Proof Idea:** Before going into the details of the complexity of $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ $(k \geq 2)$, we will look at the various ingredients required to show that $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ $(k \geq 2)$ is VNP-complete. In Section 4.1, we discuss the gadget construction of $\mathcal{H}_k$ and its properties. In Section 4.2, we discuss rosette construction and its properties. For any $f_n(X) \in \mathsf{VNP}$, we use the gadgets $\mathcal{H}_k$ and rosettes to construct the graph $T_m$ such that $\mathtt{Det}_{\mathtt{n}}^{\neq \mathtt{k}}(\mathtt{X})$ defined over $T_m$ computes $f_n(X)$.

### 4.1 Gadget Construction $\mathcal{H}_k$ and its properties

**Gadget $\mathcal{H}_k$.** For every $k > 1$, we construct a *partial iff gadget* $\mathcal{H}_k$.

- Let $V(\mathcal{H}_k) = \{a_i | 1 \leq i \leq 2k-1\}$.
- Let $E(\mathcal{H}_k) = \{(a_t, a_{t+1}) | 1 \leq t \leq 2k-2)\} \cup \{(a_1, a_{k+1}), (a_{k+1}, a_2), (a_{2k-1}, a_1)\} \cup \{(a_m, a_m) | 3 \leq m \leq 2k-1\}$. We call the edges in $E(\mathcal{H}_k)$ as the gadget edges.
- The weights of $(a_1, a_2)$ and the self-loop on $a_{k+1}$ are $-1$. All the other edge weights are 1.

Consider a directed graph $G$ with two distinct edges $(u, v)$ and $(u', v')$. We say that the gadget $\mathcal{H}_k$ is placed between the edges $(u, v)$ and $(u', v')$, if we delete both the edges $(u, v)$ and $(u', v')$ in $G$ and we add the following directed edges, $\{(u, a_1), (a_1, v), (u', a_2), (a_2, v')\}$. We set $w(u, a_1) = w(u, v)$ and $w(u', a_2) = w(u', v')$. We set $w(a_1, v) = w(a_2, v') = 1$ (see Figure 1).

Before getting into the details of the properties of the gadget $\mathcal{H}_k$, we first define *perceived sign* and *perceived monomial* of a cycle cover.

Let $G$ be a directed graph with edge labelling function $\Phi : E(G) \to X \cup \mathbb{F}$, where $X = \{x_i | 1 \leq i \leq n\}$. Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a cycle cover of $G$. Let $\{e_1, \ldots, e_t\}$ be the edges in $\mathcal{C}$. The sign of the cycle cover, which we denote as $s(\mathcal{C})$ is defined as $(-1)^{n+k}$. The monomial of $\mathcal{C}$, denoted as $m_{\mathcal{C}}$, is defined as the product of the labels of edges in $\mathcal{C}$, i.e. $\prod_{i=1}^{t} \Phi(e_i)$.

Note that the coefficient of this monomial can be either positive or negative based on the number of negatively signed edge labels in $\mathcal{C}$. We denote this sign by $s(m_{\mathcal{C}})$. Based on this, we define *perceived sign* $\hat{s}(\mathcal{C})$ and *perceived monomial* $\hat{m}_{\mathcal{C}}$ as follows.

Let $\hat{s}(\mathcal{C})$ be $(-1)^{n+k} \cdot s(m_{\mathcal{C}})$. And let $\hat{m}_{\mathcal{C}} = m_{\mathcal{C}} \cdot s(m_{\mathcal{C}})$. Note that

$$s(\mathcal{C}) \cdot m_{\mathcal{C}} = \hat{s}(\mathcal{C}) \cdot \hat{m}_{\mathcal{C}}. \tag{16}$$

**Properties of $\mathcal{H}_k$.** We now state some important properties of $\mathcal{H}_k$. Consider a directed graph $G = (V, E)$ with two distinct edges $(u, v)$ and $(u', v')$. Let $G'$ be the graph obtained by placing the gadget $\mathcal{H}_k$ between the edges $(u, v)$ and $(u', v')$ in $G$. Let $\mathcal{C}^{\neq k}$ be a cycle cover (without any cycle of length k) with *perceived sign* $\hat{s}$ which uses either both or none of the edges $(u, v)$ and $(u', v')$ in $G$, then there exists another cycle cover $\mathcal{C}'^{\neq k}$ with *perceived sign* $\hat{s'}$ in graph $G'$ such that the *perceived monomial* associated with both $\mathcal{C}^{\neq k}$ and $\mathcal{C}'^{\neq k}$ are same and $\hat{s} = \hat{s'}$. We also show that if there exists a cycle cover $\mathcal{C}^{\neq k}$ in $G$ such that it uses exactly one of the edges from $(u, v)$ and $(u', v')$, then such a cycle cover does not survive in $G'$.

1. If $\mathcal{C}^{\neq k}$ does not use any of the edges $(u, v)$ and $(u', v')$ then the $\mathcal{C}'^{\neq k}$ consists of all the cycles in $\mathcal{C}^{\neq k}$ and the gadget vertices are covered within themselves by a single cycle $(a_1, a_2, \ldots, a_{2k-1}, a_1)$. The total number of cycles in cycle cover $\mathcal{C}'^{\neq k}$ is one more than the number of cycles in cycle cover $\mathcal{C}'^{\neq k}$, but since the weight of edge $(a_1, a_2)$ is $-1$, the *perceived sign* of cycle cover $\mathcal{C}'^{\neq k}$ is same as the *perceived sign* of cycle cover $C^{\neq k}$.
2. If $\mathcal{C}^{\neq k}$ uses both the edges $(u, v)$ and $(u', v')$ then $\mathcal{C}'^{\neq k}$ has all the cycles in $\mathcal{C}^{\neq k}$ except that the edges $(u, v)$ and $(u', v')$ are now replaced by directed path $(u, a_1, v)$ and $(u', a_2, v')$ respectively. The vertices $a_3$, $a_4$, $\ldots$, $a_{2k-1}$ are covered by self-loops (of weight 1) in $\mathcal{C}'^{\neq k}$. Since the total number of cycles covered by self-loops in the gadget is always odd and the self-loop on the vertex $a_{k+1}$ has weight $-1$, the *perceived sign* is preserved.
3. If $\mathcal{C}^{\neq k}$ uses the edge $(u, v)$ but the not the edge $(u', v')$ then there is only one way to cover the vertices $a_2, a_3, \ldots, a_{k+1}$ with a cycle $(a_2, a_3, \ldots, a_{k+1}, a_2)$ of length $k$, which is not a valid cycle cover.
4. If $\mathcal{C}^{\neq k}$ uses the edge $(u', v')$ but not the edge $(u, v)$ then there is only one way to cover the vertices $a_{k+1}, a_{k+2}, \ldots, a_{2k-1}$ with a cycle $(a_{k+1}, a_{k+2}, \ldots, a_{2k-1}, a_{k+1})$ of length $k$, which is not a valid cycle cover.
5. Note that after placing the gadget $\mathcal{H}_k$, there could be new cycle covers that arise in the graph $G'$ which were not present in graph $G$. There are only two possible cases. Out of these two cases, in one of the cases, the contribution

of all such cycle covers in the overall sum is 0 whereas in the other case, the contribution is not 0. We explain both the cases in detail below.

(a) Let $\mathcal{C}^{\neq k}$ be a cycle cover consisting of a cycle, say $\mathcal{C}_1$ starting with vertex $u$ followed by the edge $(u, a_1)$ and then the edge path $\mathcal{P}_1 = (a_1, a_2)$ followed by an edge $(a_2, v')$. Since there are two paths from $a_1$ and $a_2$ (within the gadget vertices), there exists another cycle cover $\mathcal{C}'^{\neq k}$ consisting of a cycle $\mathcal{C}'_1$ starting with vertex $u$ followed by the edge $(u, a_1)$ and then the path $\mathcal{P}'_1 = (a_1 - a_{k+1} - a_2)$ followed by an edge $(a_2, v')$ such that the perceived monomials of both $\mathcal{C}^{\neq k}$ and $\mathcal{C}'^{\neq k}$ are same. Moreover, since the number of cycles in cycle cover $\mathcal{C}'^{\neq k}$ is one less than the number of cycles in cycle cover $\mathcal{C}^{\neq k}$, their perceived signs are different. In other words, there exists a bijection $\Psi$ between the set of cycle covers using $\mathcal{P}_1$ in one of its cycles and the set of cycle covers using $\mathcal{P}'_1$ in one of its cycles such that for any $\mathcal{C}$, $\Psi(\mathcal{C})$ and $\mathcal{C}$ have same perceived monomials but with opposite perceived signs. Therefore, the contribution of such cycle covers to the overall sum is 0.

(b) Let $\mathcal{C}^{\neq k}$ be a cycle cover consisting of a cycle, say $\mathcal{C}_1$ with a path $\mathcal{P}_1$ in it starting with vertex $u'$ followed by the edge $(u', a_2)$ and then a path from vertex $a_2$ to vertex $a_1$ (within the gadget $\mathcal{H}_k$) followed by an edge $(a_1, v)$. There are no cancellations possible in this case and therefore, such cycle covers survives.

*Remark 3.* Unlike the Valiant's iff gadget, in this gadget, we do not guarantee that the contribution of cycle covers (which may arise due to the placing of this gadget) is 0 (see Point 5(b) above). Therefore, we call $\mathcal{H}_k$ *partial iff gadget*. For the proof to work, we exploit the property of the graph on which these gadgets are placed such that the contribution of such new cycle covers is 0.

### 4.2 Rosette Construction $R(\ell, \mathcal{I})$

In this section, we describe the rosette construction $R(\ell, \mathcal{I})$ for every $\ell > k$ and $\mathcal{I} \subseteq [\ell]$. The construction of $R(\ell, \mathcal{I})$ is very similar to the construction of $R(\ell)$ as stated in [16], except for some modifications to incorporate the restriction about the length of the cycle in the cycle cover. Formally, we consider a directed cycle $\mathcal{C}$ of length $\ell$ with vertex set $|V(\mathcal{C})| = \{u_1, u_2, \ldots, u_\ell\}$ and $E(\mathcal{C}) = \{e_i = (u_i, u_{i+1}) | 1 \leq i \leq \ell - 1\} \cup \{e_\ell = (u_\ell, u_1)\}$. We call the edges in $|E(\mathcal{C})|$ as *connector edges* and the vertices in $|V(\mathcal{C})|$ as *connector vertices*. Consider a set $\mathcal{S}(\mathcal{I}) = \{e_i | i \in \mathcal{I}\}$. For every edge $(u_i, u_j)$ in $\mathcal{S}(\mathcal{I})$, we add a new vertex $t_{i,j}$ and add edges $(u_i, t_{i,j})$ and $(t_{i,j}, u_j)$. We call the edges in set $\mathcal{S}(\mathcal{I})$ as *participating edges*. We add self-loops on all the vertices of our graph. We arbitrarily pick one of the connector vertices and set the weight of the self-loop on it to 1, whereas the weights of all the other self-loops are set to $-1$. The weights of all the edges in rosette $R(\ell, \mathcal{I})$ (except the self-loops) are set to 1. This completes the construction of $R(\ell, \mathcal{I})$ (see Figure 2). It is easy to observe that every $R(\ell, \mathcal{I})$ contains a unique longest cycle. We denote this cycle by $\mathcal{Z}$. The rosette $R(\ell, \mathcal{I})$ satisfies the following four properties.

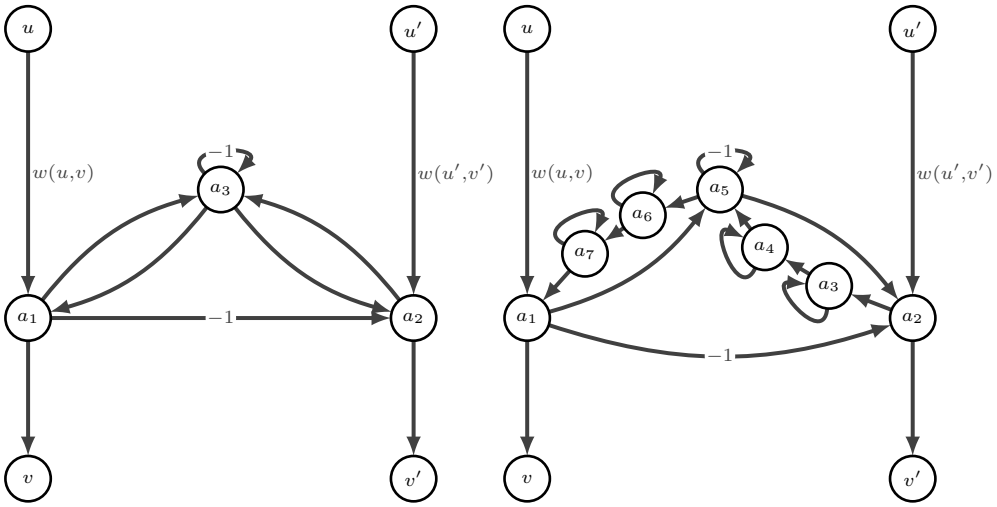


**Fig. 1.** Partial iff gadget $\mathcal{H}_2$ (left) and Partial iff gadget $\mathcal{H}_4$ (right).

1. There is no cycle cover in $R(\ell, \mathcal{I})$ that contains a cycle of length $k$.
2. For any subset $\phi \neq X \subseteq \mathcal{S}(\mathcal{I})$, there exists exactly one cycle cover of $R(\ell, \mathcal{I})$ which, among the participating edges, contains exactly the edges in $X$. Such a cycle cover always contains a single cycle which is not a loop and all other remaining vertices are covered with self loops.
3. There are only two cycle covers of $R(\ell, \mathcal{I})$ which contain no participating edges. The first cycle cover consist of only self-loops on each of the vertices in $R(\ell, \mathcal{I})$. The other cycle cover consists of the unique longest cycle $\mathcal{Z}$.
4. There are no other cycle covers in $R(\ell, \mathcal{I})$.

Note that for any cycle cover $C$ of rosette $R(\ell, \mathcal{I})$, the *perceived sign* is $-1$ and the *perceived monomial* is 1.

### 4.3 Construction of graph $T_m$ from $f_n(X) \in \mathsf{VNP}$

Let $f_n(X) \in \mathsf{VNP}$. From the definition of $\mathsf{VNP}$ we know that $f_n(X) = \sum_{y_1, y_2, \ldots, y_{p(n)} \in \{0,1\}} g'_n(X, Y)$, where $p : \mathbb{N} \longrightarrow \mathbb{N}$ is polynomially bounded function in $n$ and $g'_n(X, Y)$ is in $\mathsf{VP}$. Moreover, for any $f_n(X) \in \mathsf{VNP}$, $f_n(X) = \sum_{y_1, y_2, \ldots, y_{p(n)} \in \{0,1\}} g_n(X, Y)$, where $g_n(X, Y)$ is in $\mathsf{VF}$ [13].

1. Since, $g_n(X, Y) \in \mathsf{VF}$ and $\mathsf{VF} \subseteq \mathsf{VBP}$, there exists an algebraic branching program of size $s = \text{poly}(n)$ to compute $g_n(X, Y)$, say $\mathcal{B}_n$. Let $s_0$ and $t_0$ be the source and sink of $\mathcal{B}_n$, respectively. Without loss of generality assume that the length of the longest path from $s_0$ to $t_0$ in $\mathcal{B}_n$ is at least $k$. We add a special vertex $\alpha$ and add directed edges from $\alpha$ to $s_0$ and from $t$ to $\alpha$. We set the weights of both the edges $(\alpha, s_0)$ and $(t_0, \alpha)$ to 1. We add self-loops

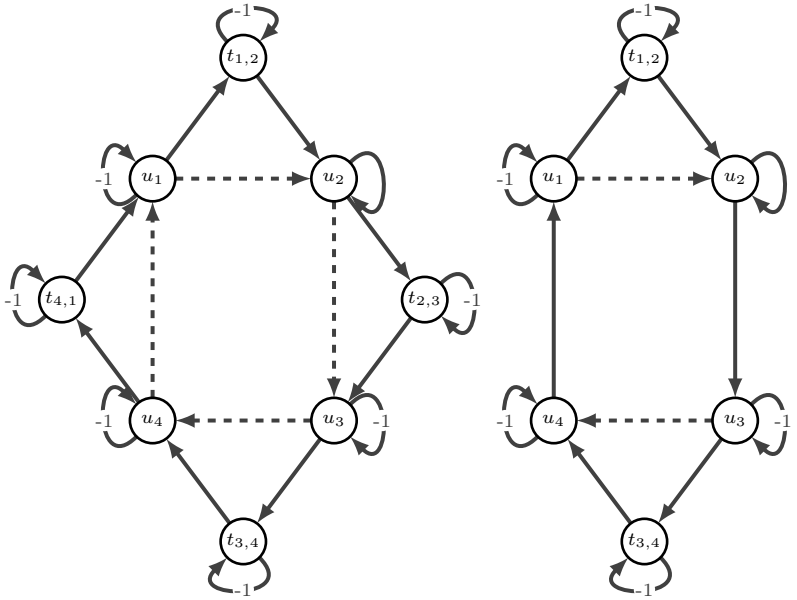


**Fig. 2.** $R(4, \mathcal{I})$, where $\mathcal{S}(\mathcal{I}) = \{e_1, e_2, e_3, e_4\}$ (left) and $R(4, \mathcal{I})$, where $\mathcal{S}(\mathcal{I}) = \{e_1, e_3\}$ (right). Dashed edges are participating edges.

on all the vertices in our constructed graph except $\alpha$. We set the weight of all self-loops to be 1. We call this graph $\hat{\mathcal{B}}_n$.

2. We know that $Y = \{y_i | 1 \leq i \leq p(n)\}$. Let $occ(i)$ denote the total number of edges in $\hat{\mathcal{B}}_n$ which are labelled with variable $y_i$. Let $Y_1 = \{y_i | y_i \in Y, occ(i) > k\}$ and $Y_2 = \{y_i | y_i \in Y, occ(i) \leq k\}$. Cleary, $Y = Y_1 \cup Y_2$. For every $y_i \in Y_1$, we consider a rosette $R(occ(i), \mathcal{I} = [occ(i)])$. Similary, for every $y_j \in Y_2$, we consider a rosette $R(occ(j) + k, \mathcal{I} \subset [occ(j)] + k)$, where the set $|\mathcal{S}(\mathcal{I})| = occ(j)$. We call the graph constructed so far as the *partial graph* denoted by $\tilde{T}_m$.
3. For any $y_i \in Y_1$, let $c_{y_i,1}, c_{y_i,2}, \ldots, c_{y_i,occ(i)}$ be the connector edges in $R(occ(i), \mathcal{I} = [occ(i)])$. Let $e_{y_i,1}, e_{y_i,2}, \ldots, e_{y_i,occ(i)}$ be the distinct edges in $\hat{\mathcal{B}}_n$ which are labelled with variable $y_i$. We place $\mathcal{H}_k$ between the edges $c_{y_i,t}$ and $e_{y_i,t}$ for all $1 \leq t \leq occ(i)$.
4. For any $y_j \in Y_2$, let $c_{y_j,1}, c_{y_j,2}, \ldots, c_{y_j,occ(j)}, \ldots, c_{y_j,occ(j)+k}$ be the connector edges in $R(occ(j) + k, \mathcal{I} \subset [occ(j) + k])$. Without loss of generality, let us assume that $\mathcal{S}(\mathcal{I}) = \{c_{y_j,1}, c_{y_j,2}, \ldots, c_{y_j,occ(j)}\}$. Let $e_{y_j,1}, e_{y_j,2}, \ldots, e_{y_j,occ(j)}$ be the distinct edges in $\hat{\mathcal{B}}_n$ which are labelled with variable $y_j$. We place "partial iff gadget" $\mathcal{H}_k$ between the edges $c_{y_j,t}$ and $e_{y_j,t}$ for all $1 \leq t \leq occ(j)$.

This completes the construction of graph $T_m$. It is easy to note that $m = \text{poly}(n)$.

### 4.4 Cycle covers of graph $\hat{\mathcal{B}}_n$

In this section, we study the cycle covers of graph $\hat{\mathcal{B}}_n$. Recall that the construction of graph $\hat{\mathcal{B}}_n$ is stated in the first point of Section 4.3.

**Lemma 9.** *Let $\mathcal{B}_n$ be the algebraic branching program of size $s = \text{poly}(n)$ computing $g_n(X, Y)$. Let $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq \mu\}$ be the set of all $s - t$ paths in $\mathcal{B}_n$. Let $m_i$ be the monomial associated with path $\mathcal{P}_i$, formed by multiplying the labels on all the edges of $\mathcal{B}_n$ participating in path $\mathcal{P}_i$.*

- *For each $i \in [\mu]$, there exists a unique cycle cover $C$ in $\hat{\mathcal{B}}_n$, such that the monomial associated with $C$ is $m_i$. Moreover, there are no other cycle covers in $\mathcal{B}_n$.*
- *The number of cycles in any cycle cover of graph $\hat{\mathcal{B}}_n$ is same.*

*Proof.* Since, the vertex $\alpha$ is not covered by a self-loop, the only way to cover vertex $\alpha$ in $\hat{\mathcal{B}}_n$ is by a cycle starting form the vertex $\alpha$ followed by the edge $(\alpha, s)$, followed by some directed path in $\hat{\mathcal{B}}_n$ followed by the edge $(t, \alpha)$. Let $\mathcal{C}_j$ be a cycle cover of graph $\hat{\mathcal{B}}_n$. Let $\mathcal{C}_j$ consists of a cycle $\mathcal{C}_{j_1}$ which starts at vertex $\alpha$ and traverses the edge $(\alpha, s)$, followed by a $s-t$ directed path $\mathcal{P}_j$ (in algebraic branching program $\mathcal{B}_n$) in the graph $\hat{\mathcal{B}}_n$ followed by the edge $(t, \alpha)$. Note that all the remaining vertices in $\hat{\mathcal{B}}_n$ must get covered by self-loops in cycle cover $\mathcal{C}$. Note that the weight of all the self-loops along with edges $(s, \alpha)$ and $(t, \alpha)$ is set to 1 and by the the construction of graph $\hat{\mathcal{B}}_n$, it is easy to follow that the monomial associated with $\mathcal{P}_j$ is $m_j$. It is easy to observe that there are no other cycle covers in graph $\hat{\mathcal{B}}_n$. It is easy to note that the total number of cycles in any cycle cover of graph $\hat{\mathcal{B}}_n$ is same. □

### 4.5 Surviving cycle covers of graph $\tilde{T}_m$ in graph $T_m$

In this section, we study the surviving cycle covers of graph $\tilde{T}_m$ in graph $T_m$. Recall that the construction of graph $\tilde{T}_m$ is stated in the second point of Section 4.3.

We know that $\tilde{T}_m$ denotes the partial graph which is the disjoint union of $\mathcal{B}_n$ and $p(n)$ disjoint copies of rosette for each $y_i \in Y$. The cycle cover $\mathcal{C}$ of $\tilde{T}_m$ is called a surviving cycle cover if it survives in $T_m$. We now carefully analyze the surviving cycle cover of $\tilde{T}_m$. Consider the *partial graph* $\tilde{T}_m$. It is easy to note that any cycle cover $\mathcal{C}$ of graph $\tilde{T}_m$ is the union of the cycle cover of each of its components. Let $\mathcal{C}_j$ be a cycle cover of graph $\hat{\mathcal{B}}_m$. Let $\mathcal{C}_j$ consist of a cycle $\mathcal{C}_{j_1}$ which starts at vertex $\alpha$ and traverses the edge $(\alpha, s_0)$, followed by a $s_0 - t_0$ directed path $\mathcal{P}_j$ (in the algebraic branching program $\mathcal{B}_n$) in the graph $\hat{\mathcal{B}}_n$ followed by the edge $(t_0, \alpha)$. Note that all the remaining vertices in $\hat{\mathcal{B}}_n$ must get covered by self-loops in cycle cover $\mathcal{C}$. By Lemma 9, let $m_j$ be the monomial associated with $\mathcal{C}$ formed by multiplying all the labels of edges participating in $\mathcal{C}$.

Let $\mathcal{T}_j \subseteq Y$ be the set of variables whose degree is at least 1 in $m_j$. Let $R_1, \ldots R_{p(n)}$ be the rosettes associated with variables $y_1, \ldots y_{p(n)}$, respectively in graph $\tilde{T}_m$. Without loss of generality, let $\mathcal{T}_j = \{y_1, y_2, y_3\}$, where

the degrees of $y_1, y_2, y_3$ in $m_j$ are given by $deg_{y_1}, deg_{y_2}, deg_{y_3}$, respectively. Let $e_{y_1,1}, e_{y_1,2}, \ldots, e_{y_1,deg_{y_1}}$ be the distinct edges labelled with $y_1$ in path $\mathcal{P}_j$. Similarly, let $e_{y_2,1}, e_{y_2,2}, \ldots, e_{y_2,deg_{y_2}}$ be the distinct edges labelled with $y_2$ in path $\mathcal{P}_j$. Also, let $e_{y_3,1}, e_{y_3,2}, \ldots, e_{y_3,deg_{y_3}}$ be the distinct edges labelled with $y_3$ in path $\mathcal{P}_j$. Let $\mathcal{R}_1, \ldots \mathcal{R}_{p(n)}$ be the cycle covers in the rosettes $R_1, \ldots R_{p(n)}$, respectively. Let us assume that the cycle covers $\mathcal{R}_1, \ldots \mathcal{R}_{p(n)}$ satisfies the following properties:

1. For each $k \in [3]$, the cycle cover $\mathcal{R}_k$ in rosette $R_k$ uses the participating edges
   $c_{y_k,1}, c_{y_k,2}, \ldots, c_{y_k,deg_{y_k}}$ and no other participating edges in $R_k$.
2. For each $k \in [p(n)] \backslash [3]$, the cycle cover $\mathcal{R}_k$ does not use any participating edges of $R_k$.

For $k \in [p(n)]$, we know that there exists a "partial if and only if gadget" between $e_{y_k,i}$ and $c_{y_k,i}$ for all $1 \leq i \leq occ(k)$ in $T_m$. By the property of the "partial if and only if" gadget, the cycle covers described in the above case will survive in graph $T_m$, whereas all the other cycle covers will not survive in graph $T_m$.

### 4.6 Proof of VNP completeness

In this section, we state our main proof.
Let $f_n(X) \in \mathsf{VNP}$. Let $\tilde{T}_m$ and $T_m$ denote the graphs constructed from $f_n(X)$ as stated in Section 4.3. We express $g_n(X, Y)$ as the sum of monomials.
We know that there exists an algebraic branching program $\mathcal{B}_n$ which computes the polynomial $g_n(X, Y)$. Let the set $\mathcal{P}$ consist of all the $s - t$ path in $\mathcal{B}_n$. Let $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq \mu\}$. Let $c_i m_i$ denotes the monomial associated with path $\mathcal{P}_i$ in $\mathcal{B}_n$, where $c_i \in \mathbb{F}$ is the coefficient of $m_i$. Therefore, $g_n(X, Y) = c_1 m_1 + c_2 m_2 + \ldots + c_\mu m_\mu$,

We have, $f_n(X) = \sum_{y_1,y_2,\ldots,y_{p(n)} \in \{0,1\}} c_1 m_1 + c_2 m_2 + \ldots + c_\mu m_\mu$

$= \sum_{y_1,y_2,\ldots,y_{p(n)} \in \{0,1\}} c_1 m_1 + \sum_{y_1,y_2,\ldots,y_{p(n)} \in \{0,1\}} c_2 m_2 + \ldots + \sum_{y_1,y_2,\ldots,y_{p(n)} \in \{0,1\}} c_\mu m_\mu$.

Let $\mathcal{J}_j$ denotes the polynomial $\sum_{y_1,y_2,\ldots,y_{p(n)}} c_j m_j$. Therefore, $f_n(X) = \sum_{j=1}^{\mu} \mathcal{J}_j$.

We know that $m_j$ is a monomial over the variable set $X \cup Y$. Therefore, $m_j$ can be represented as the product of two monomials $m_j^{(X)}$ and $m_j^{(Y)}$, where $m_j^{(X)}$ is over the variable set $X$ and $m_j^{(Y)}$ is over the variable set $Y$.
That is,

$g_n(X, Y) = c_1 m_1^{(X)} m_1^{(Y)} + c_2 m_2^{(X)} m_2^{(Y)} + \ldots + c_\mu m_\mu^{(X)} m_\mu^{(Y)}$.

Consider a monomial $m_j = c_j m_j^{(X)} m_j^{(Y)}$, let $\mathcal{T}_j \subseteq Y$ is the set of variables whose degree is at least 1 in $m_j^{(Y)}$. Let $m_j(a_1, a_2, \ldots, a_{p(n)})$ be the evaluation of monomial $m_j$ at point $y_1 = a_1, y_2 = a_2, \ldots, y_n = a_n$. For example, let us assume that $\mathcal{T}_j = \{y_1, y_2\}$. (The case when $\mathcal{T}_j$ has higher cardinality is very similar.) An evaluation of monomial $m_j$ at any point where at least one variable from $\mathcal{T}_j \subseteq Y$ is set to 0 vanishes. The evaluations where the monomial $m_j$ survives is only at the points of the form $y_1 = 1, y_2 = 1, y_3 = *, \ldots, y_{p(n)} = *$, where $* \in \{0, 1\}$. That is, the coefficient of $c_j m_j^{(X)}$ is equal to the $2^{p(n)-|\mathcal{T}_j|}$ in $\mathcal{J}_j$. This is true for every monomial $c_j m_j$, for all $1 \leq j \leq \mu$.

We consider two cases in this proof. In the first case, we show that for every monomial $c_j m_j$ evaluated at $y_1 = a_1, y_2 = a_2, y_3 = a_3, \ldots, y_{p(n)} = a_{p(n)}$ (as nonzero), there exists a unique cycle cover $\mathcal{C}$ (without any cycles of length $k$) in graph $T_m$ such that the product of perceived monomial and perceived sign associated with $\mathcal{C}$ is equal to $c_j m_j$ evaluated at $y_1 = a_1, y_2 = a_2, y_3 = a_3, \ldots, y_{p(n)} = a_{p(n)}$. In the second case, we show that there are no other cycle covers in graph $T_m$.

Before going into the details of the cases, we define a partial cycle cover and an extension of a cycle cover.

**Definition 12.** *Let $G = (V, E)$ be a directed graph. A partial cycle cover $\mathcal{C}'$ of $G$ over vertex set $V' \subset V$ is a set of subgraphs of $G$ which are cycle graphs such that every vertex $v' \in V'$ participates in exactly one of the cycles in $\mathcal{C}'$. We say that a cycle cover $\mathcal{C}$ (of graph $G$) is an extension of a partial cycle cover (of graph $G$) $\mathcal{C}'$ if and only if $\mathcal{C}' \subset \mathcal{C}$.*

**Case 1:** Fix $j$. Let $\mathcal{C} = \mathcal{C}_j$ be the partial cycle cover of graph $\tilde{T}_m$, where $\mathcal{C}_j$ is a cycle cover of graph $\hat{\mathcal{B}}_m$. Let $m_j$ be the monomial associated with $\mathcal{C}_j$. Let $\mathcal{T}_j \subseteq \{y_1, y_2, y_3\}$ is the set of variables whose degree is at least 1 in $m_j$.

As stated in Section 4.5, for any extension of cycle cover $\mathcal{C}$ in graph $\tilde{T}_m$ to survive, the cycle covers of rosettes $R_1, R_2, \ldots, R_{p(n)}$ must satisfy the following properties:

1. For each $k \in [3]$, the cycle cover $\mathcal{R}_k$ in rosette $R_k$ uses the participating edges
   $c_{y_k,1}, c_{y_k,2}, \ldots, c_{y_k,deg_{y_k}}$ and no other participating edges in $R_k$. By property 1 of the rosettes, this can be done in only one way.
2. For each $k \in [p(n)] \backslash [3]$, the cycle cover $\mathcal{R}_k$ does not use any participating edges of $R_k$. By property 2, this can be done in two ways.

Therefore, there are exactly $2^{p(n)-3}$ extensions possible to the cycle cover $\mathcal{C}$. In general there are exactly $2^{p(n)-|\mathcal{T}_j|}$ extensions possible to the cycle cover $\mathcal{C}$. This is true for every $1 \leq j \leq \mu$. Note that there are no other cycle covers possible in $\tilde{T}_m$. .

**Case 2:** Recall that there could be new cycle covers which arise in our graph $T_m$ after placing the gadgets as described in point 5(b) of the properties of gadget $\mathcal{H}_k$. We exploit the properties of graph $\tilde{T}_m$ on which the gadgets are placed and show that the contribution of such cycle covers is also 0. Consider the graph $T_m$. Let us assume that there exists a cycle cover $\mathcal{C}'$ which is not of the form discussed in case 1. Let us assume that $\mathcal{C}'$ is a cycle cover consisting of a cycle $\mathcal{C}_1$ which hits a vertex $a_2$ in some copy of the "partial if and only if gadget" $\mathcal{H}_k$ and uses the gadget edges hereafter to reach vertex $a_1$ of the same gadget. Such a cycle ($\mathcal{C}_1$ in this case) will always have a path which goes from vertex $a_1$ to $a_2$ of some other copy of gadget $\mathcal{H}_k$.Note that there are exactly two distinct paths in the "partial if and only if gadget" $\mathcal{H}_k$ to reach from $a_1$ to $a_2$. Therefore, for such a cycle cover $\mathcal{C}'$, there always exists another cycle cover $\widehat{\mathcal{C}}$, such that the perceived monomials associated with both cycle covers $\mathcal{C}'$ and $\widehat{\mathcal{C}}$ are same but with opposite perceived signs.

## References


1. Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *computational complexity*, 25(1):217–253, 2016.
2. Dario Bini. Relations between exact and approximate bilinear algorithms. applications. *Calcolo*, 17(1):87–97, 1980.
3. Dario Bini et al. O($n^{2.7799}$) complexity for $n \times n$ approximate matrix multiplication. 1979.
4. Markus Bläser, Christian Ikenmeyer, Meena Mahajan, Anurag Pandey, and Nitin Saurabh. Algebraic branching programs, border complexity, and tangent spaces. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 21:1–21:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
5. Andreas Brandstädt, Van Bang Le, and Jeremy P Spinrad. *Graph classes: a survey*. SIAM, 1999.
6. Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *Journal of the ACM (JACM)*, 65(5):1–29, 2018.
7. Prasad Chaugule, Nutan Limaye, and Aditya Varre. Variants of homomorphism polynomials complete for algebraic complexity classes. In *Computing and Combinatorics - 25th International Conference, COCOON*, volume 11653 of *LNCS*, pages 90–102. Springer, 2019.
8. Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism polynomials complete for VP. *Chic. J. Theor. Comput. Sci.*, 2016, 2016.
9. Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
10. Christian Ikenmeyer and Abhiroop Sanyal. A note on VNP-completeness and border complexity. *Information Processing Letters*, 176:106243, 2022.

11. Mrinal Kumar. On the power of border of depth-3 arithmetic circuits. *ACM Transactions on Computation Theory (TOCT)*, 12(1):1–8, 2020.
12. Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chic. J. Theor. Comput. Sci.*, 1997, 1997.
13. Guillaume Malod and Natacha Portier. Characterizing valiant's algebraic complexity classes. In *MFCS*, pages 704–716. Springer, 2006.
14. Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory I: An approach to the P vs. NP and related problems. *SIAM Journal on Computing*, 31(2):496–526, 2001.
15. Ran Raz. Elusive functions and lower bounds for arithmetic circuits. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 711–720. ACM, 2008.
16. Nitin Saurabh. Algebraic models of computation. *MS Thesis*, 2012.
17. Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5(3–4):207–388, mar 2010.
18. 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.
19. L. G. Valiant. Completeness classes in algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 249–261, 1979.

**Acknowledgement:** We would like to thank Shourya Pandey, Radu Curticapean, and the anonymous reviewers who gave useful comments on the earlier draft of the paper.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il