Exponential lower bounds via exponential sums

Somnath Bhattacharjee
Chennai Mathematical Institute, Chennai, India

Markus Bläser
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Pranjal Dutta
School of Computing, National University of Singapore

Saswata Mukherjee
Chennai Mathematical Institute, Chennai, India

Abstract

Valiant's famous VP vs. VNP conjecture states that the symbolic permanent polynomial does not have polynomial-size algebraic circuits. However, the best upper bound on the size of the circuits computing the permanent is exponential. Informally, VNP is an exponential sum of VP-circuits. In this paper we study whether, in general, exponential sums (of algebraic circuits) *require* exponential-size algebraic circuits. We show that the famous Shub-Smale τ-conjecture indeed implies such an exponential lower bound for an exponential sum. Our main tools come from parameterized complexity. Along the way, we also prove an exponential fpt (fixed-parameter tractable) lower bound for the parameterized algebraic complexity class $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}]$, assuming the same conjecture. $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}]$ can be thought of as the weighted sums of (unbounded-degree) circuits, where only ± 1 constants are *cost-free*. To the best of our knowledge, this is the *first* time the Shub-Smale τ-conjecture has been applied to prove explicit exponential lower bounds.

Furthermore, we prove that when this class is fpt, then a variant of the counting hierarchy, namely the *linear counting hierarchy* collapses. Moreover, if a certain type of parameterized exponential sums is fpt, then integers, as well as polynomials with coefficients being *definable* in the linear counting hierarchy have subpolynomial τ-complexity.

Finally, we characterize a related class VW[F], in terms of permanents, where we consider an exponential sum of algebraic formulas instead of circuits. We show that when we sum over cycle covers that have one long cycle and all other cycles have constant length, then the resulting family of polynomials is *complete* for VW[F] on certain types of graphs.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Algebraic complexity, parameterized complexity, exponential sums, counting hierarchy, tau conjecture

1 Introduction

Valiant [26] proposed an algebraic version of the P versus NP question and defined the class VP, the algebraic analogue of P, which contains polynomial families computable by polynomial sized algebraic circuits. An *algebraic circuit* (or, arithmetic circuit) C is a directed acyclic graph such that (1) every node has either in-degree (*fan-in*) 0 (the *input gates*) or 2 (the *computational gates*), (2) every input gate is labeled by elements from a field $\mathbb{K}$ or variables from $\mathbf{X} = \{X_1, \cdots, X_n\}$, (3) every computational gate is labeled by either $+$ (addition gate) or $\times$ (multiplication gate), with the obvious syntactic meaning, and (4) there is a unique gate of out-degree 0, the *output gate*. Clearly, every gate in a circuit computes a polynomial in $\mathbb{K}[\mathbf{X}]$. We say that the circuit C computes $P(\mathbf{X}) \in \mathbb{K}[\mathbf{X}]$ if the output gate of C computes $P(\mathbf{X})$. The *size* of C, denoted by size(C), is the number of nodes in the circuit. An algebraic circuit is an *algebraic formula* if every gate in the circuit has out-degree 1 except for the output gate. The class VNP, the algebraic analogue of NP, is definable by

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:32
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

taking *exponential sums* of the form

$$f(\mathbf{X}) = \sum_{e\in\{0,1\}^\ell} g(\mathbf{X}, e)\ , \tag{1}$$

where g is computable by a polynomial-size circuit and ℓ is polynomial in the number of variables. It is known that one can also replace algebraic circuits by algebraic formulas, and still get the same class VNP [26, 21]. Valiant further proved that the permanent family is complete for VNP (over fields of characteristic not two). Recall that the permanent of a matrix $(X_{i,j})$ is defined as

$$\operatorname{per}\mathbf{X} = \sum_{\pi\in S_n} X_{1,\pi(1)}\cdots X_{n,\pi(n)}. \tag{2}$$

The famous Valiant's conjecture $\mathsf{VP}\neq\mathsf{VNP}$ is equivalent to the fact that the permanent does not have polynomial-size circuits. The representation of the permanent in (2), although it looks very natural, is not *optimal*. Ryser's formula [22] yields an algebraic formula of size $O(2^n n^2)$. A formula of similar size was later found by Glynn [14]. Ryser's formula is now over sixty years old and has not been improved since. This gives rise to the interesting question whether there is a formula or circuit of subexponential-size (in n) for the permanent? More generally, we can now ask the following question.

► **Question 1.** *Is an exponential sum f (as in Eq. (1)) always computable by an algebraic circuit or formulas of size subexponential in ℓ, that is, size $2^{o(\ell)}$? Or are there instances for which exponential-size is necessary?*

Note that exponential-size being necessary is a much *stronger* claim than $\mathsf{VP}\neq\mathsf{VNP}$. It could well be that $\mathsf{VP}\neq\mathsf{VNP}$ but still exponential sums like in (1) have subxponential size circuits! In this paper, we shed some light on the question what happens if exponential sums would always have *subexponential* size circuits.

Question 1 works as driving force between the famous Shub-Smale τ-conjecture [23] and *exponential* lower bounds on exponential sums. The τ-complexity $\tau(f)$ of an integer polynomial is the size of a smallest division-free circuit that computes f starting from the constants ± 1. The *τ-conjecture* states the the number of integer zeroes of f is polynomially bounded in $\tau(f)$, see [23]. [23] shows that the τ-conjecture implies $\mathsf{P}_{\mathbb{C}}\neq\mathsf{NP}_{\mathbb{C}}$, in the Blum–Shub–Smale (BSS) model of computation over the complex numbers [5, 4].

Super-polynomial lower bounds assuming the τ-conjecture. Bürgisser [8] connected the τ-complexity of the permanent to various other conjectures. He showed that the τ-conjecture implies a *superpolynomial* lower bound on $\tau(\operatorname{per}_n)$, implying the constant-free version of $\mathsf{VP}\neq\mathsf{VNP}$, namely $\mathsf{VP}^0\neq\mathsf{VNP}^0$; for definitions, see Section 2.1. The proof strategy of [8] is as follows: assume $\tau(\operatorname{per}_n)=\mathsf{poly}(n)$, and conclude a complexity-theoretic 'collapse' that the counting hierarchy CH (for a definition, see Section 3) is in P/poly. Consider the Pochhammer–Wilkinson polynomial $f_n(x):=\prod_{i=1}^n(x-i)$, and construct a unique $O(\log n)$-variate multilinear polynomial B_n such that under a 'suitable' substitution, one gets back f_n. The coefficients of f_n as well as B_n, are efficiently computable (since $\mathsf{CH}\subseteq\mathsf{P}/\mathsf{poly}$), implying $B_n\in\mathsf{VNP}^0$. An inspection of Valiant's completeness result reveals that if $B_n\in\mathsf{VNP}^0$, then there is a polynomially bounded sequence $p(n)$ such that $\tau(2^{p(n)}B_n)=\mathsf{poly}(\log n)$, which implies $\tau(2^{p(n)}f_n)=\mathsf{poly}(\log n)$, contradicting the τ-conjecture.

In [8], the superpolynomial lower bound on $\tau(\operatorname{per}_n)$ was also implied by any of the quantities $\tau(n!)$, $\tau(\sum_{k=0}^n \frac{1}{k!}T^k)$, or $\tau(\sum_{k=0}^n k^r T^k)$ (for any fixed negative integer r) not being poly-logarithmically bounded as a function of n. Here, we remark that the separation proof of

VP^0 and VNP^0, even assuming *strong* bounds on the τ-conjecture, is merely *superpolynomial*: we *do not* get the (possibly) desirable exponential separation between VP^0 and VNP^0. This leads to the following question.

▶ **Question 2.** *Does the τ-conjecture imply exponential algebraic lower bounds?*

Here, we mention that there are variants of the τ-conjecture, e.g., the *real τ-conjecture* [18, 24] or *SOS-τ-conjecture* [11], which also give strong algebraic lower bounds. There is also super polynomial lower bound known from a proof complexity theoretic view due to [1] from the original Shub-Smale τ-conjecture. However, the Shub-Smale τ-conjecture is *not known* to give an exponential lower bound for the permanent.

1.1 Our results

The results of our paper revolve around answering both Question 1-2 positively. The main result is the following.

▶ **Theorem 1** (Informal). *The τ-conjecture implies an exponential lower bound for some explicit exponential sum.*

Remarks. (1) Although the existence of *some* polynomial requiring exponential circuits is clear from dimension/counting, the existence of an (even non-explicit) exponential sum polynomial requiring exponential-size circuits is *unclear*. Explicit here means that the family is in VNP.

(2) One can also think of an exponential sum f in Equation (1), as $f = \sum_{e\in\{0,1\}^{\ell(n)}} U(\mathbf{X}, y, e)$, where $U(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a *universal circuit* of size $\text{size}(U) = \mathsf{poly}(\text{size}(g))$ with $\mathbf{Y} = (Y_1, \ldots, Y_r)$ and $\mathbf{Z} = (Z_1, \ldots, Z_{\ell(n)})$ and $y \in \mathbb{F}^r$ is chosen such that $U(\mathbf{X}, y, e) = g(\mathbf{X}, e)$; and the number of variables $\ell(n)$ is linear in n.

(3) Since there's a polynomial (non-linear) blowup in the reduction of the exponential sum on the universal circuit from the permanent, we will only get a subexponential lower bound on the permanent polynomial assuming the τ-conjecture. We leave it as an open question to achieve an exponential lower bound on the permanent assuming the τ-conjecture.

The proof of Theorem 1 is rather indirect, and goes via *exponential sums*, which is our main object of study (and bridge between many results and classes).

log-variate exponential sum polynomial. Let $g(\mathbf{X}, \mathbf{Y})$ be some polynomial in n-many $\mathbf{X}$-variables and $\ell(n)$-many $\mathbf{Y}$-variables, where $\ell(n) = O(n)$. Assume that g is computed by a circuit of size m. Then we define

$$\mathsf{p\text{-}log\text{-}Expsum}_{m,k}(g) \;:=\; \sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)\;,$$

where $k = n/\log m$. The size of the exponential is measured in the number $\ell(n)$ of $\mathbf{Y}$-variables. In the end, we want to measure in the input size, the number n of $\mathbf{X}$-variables. To talk about subexponential complexity, $\ell(n)$ should be linearly bounded. g will be typically computed by a circuit (of unbounded degree). We want to view $\mathsf{p\text{-}log\text{-}Expsum}_{m,k}$ as a parameterized problem, the parameter will be k. Our definition of $\mathsf{p\text{-}log\text{-}Expsum}$, as a polynomial-sum, is motivated by the *log-parameterizations* which are used in the definition of the so-called M-hierarchy in the Boolean setting, see [12, 13].

We show that $\mathsf{p\text{-}log\text{-}Expsum}$ is most likely *not* fixed-parameter tractable (fpt). A polynomial family $p_{n,k}$ is fpt if both its size and degree are fpt bounded, i.e., of the form $f(k)q(n)$, for $q \le \mathsf{poly}(n)$, and $f : \mathbb{N} \to \mathbb{N}$ being *any* computable function. We connect $\mathsf{p\text{-}log\text{-}Expsum}$

with – (1) a linear variant of the counting hierarchy (we denote it by CH_{lin}), where the size of the oracle calls are bounded linearly in the size of the input; for definition see Section 3; and (2) integers definable in CH_{lin}, similar to Bürgisser [8]. Informally, an integer is definable in CH_{lin}, if its sign and bits are computable in the same class.

▶ **Theorem 2** (Informal). *If* p-log-Expsum *is fixed-parameter tractable, then the following results hold.*

1. *The linear counting hierarchy (*CH_{lin}*) collapses.*
2. *Any sequence $a(n)$ definable in the linear counting hierarchy, as well as univariate polynomials with coefficients being definable in the linear counting hierarchy, have subpolynomial τ-complexity.*

For formal statements, see Theorem 13 and 21.

Finally, many algebraic complexity classes can be defined in terms of permanents. Most prominently, the "regular" permanent family (per_n) is complete for VNP. The class VW[1] is an important class in parameterized complexity. It is defined as a bounded sum over constant depth weft-1 circuits. Bounded sum means that we sum over $\{0,1\}$-vectors with k ones and k is the parameter. Bläser and Engels [3] prove that VW[1] is described by so-called k-permanents with k being the parameter. In a k-permanent, we only sum over permutations with $n-k$ self-loops. The crucial parameterized class of this work is VW[P]: it is defined as a bounded exponential sum over polynomially-sized arithmetic circuits computing a polynomial of degree that is polynomially bounded. While we do not characterise VW[P] in terms of permanents, we characterize the related class VW[F]: Here instead of summing over circuits, we sum over *formulas.*[1] The permutations that we sum over for defining our permanent family will have one cycle of length k and all other cycles bounded by 4. Again, k is the parameter. We call the corresponding polynomials $(k,4)$-restricted permanents. It turns out that we also need to restrict the graph classes. We call a graph $G=(V,E)$ $(4,b)$-nice if we can partition the set $V=V_1\cup V_2$ disjointly, such that in the induced graph $G[V_1]$, every cycle is either a self-loop or has length >4 and in the induced graph $G[V_2]$ has tree-width bounded by b. While this looks artificial at a first glance, it turns out that there is a constant b such that $(k,4)$-restricted permanent on $(4,b)$-nice graphs describes the natural class VW[F]. There is a family of $(4,b)$-nice graphs such that the corresponding family of $(k,4)$-restricted permanents is VW[F]-hard (Corollary 41). On the other hand, the $(k,4)$-restricted permanent family is in VW[F] for every family of $(4,b)$-nice graphs (Theorem 40). Together, this implies:

▶ **Theorem 3** (VW[F]-Completeness). *$(k,4)$-restricted permanent family on $(4,b)$-nice graphs are* VW[F]*-complete.*

We also prove strong separations of algebraic complexity classes and parameterized algebraic complexity classes (Theorem 30), and exponential lower bound in the parameterized setting (Theorem 35).

For VNP it is known that it does not matter whether we sum over formulas or circuits, that is, $\mathsf{VNP}=\mathsf{VNP}_e$. Whether $\mathsf{VW[P]}=\mathsf{VW[F]}$ remains an open questions for future research.

[1] Maybe an explanation of the naming convention is helpful: In VW[P], we sum of polynomial-size circuits, which describe the class VP. In VW[F], we sum over polynomial size formulas, which define the class VF, the modern name for VP_e.

1.2 Proof ideas

In this section, we briefly sketch the proof ideas. We first present the proofs of Theorem 2, because the techniques and lemmas involved in proving them are the backbone of Theorem 1.

Proof idea of Theorem 2. We prove them in two parts.

Proof of Part (1): We prove even a stronger statement for the subexponential version of the linear counting hierarchy. The proof goes via induction on the level of the counting hierarchy. The criteria for some language B being in the $(k+1)$-th level is that there should be some language A in the k-th level such that $|\{y \in \{0,1\}^n : \langle x, y\rangle \in A\}| > 2^{n-1}$. Essentially, for a language A in the k-th level, we express $|\{y \in \{0,1\}^n : \langle x, y\rangle \in A\}| > 2^{n-1}$ as an exponential sum over an algebraic circuit $\chi_A(x, y)$, which captures the characteristic function of A. Furthermore, one can show that $\mathsf{p\text{-}log\text{-}Expsum}$ is fpt (in an unbounded constant-free setting) iff $\sum_y g(\mathbf{X}, y)$ has $2^{o(n)}\mathsf{poly}(m)$ size circuits, where g has a circuit of size m; see Theorem 15 and 16. Putting these together, one gets that the exponential sum has a subexponential-size constant-free circuit. Lastly, we want to get the information about the highest bit of the sum (which is equivalent to looking at it mod 2^n), which can be efficiently *arithmetized*. In every step there is polynomial blowup in the size, and hence the size remains subexponential, yielding the desired result. For details, see Theorem 13.

Proof of Part (2): This proof is an adaption of [8, 17] in our context. Take a sequence $(a_n)_n \in \mathsf{CH_{lin}P}$. We define a multilinear polynomial $A(\mathbf{Y})$ such that the coefficient of $\mathbf{Y}^{\mathbf{j}}$ is the j-th bit of $a(n)$, where $\mathbf{j}$ is the binary representation of j. Furthermore, checking $a(n, j) = b$ can be done by a subexponential circuit $C(\mathbf{N}, \mathbf{J})$, where $\mathbf{N}$ and $\mathbf{J}$ have $\log n$ and $\mathsf{bit}(n)$-many variables capturing n and j respectively. Moreover, one can define $F(\mathbf{N}, \mathbf{Y}, \mathbf{J}) = C(\mathbf{N}, \mathbf{J}) \cdot \prod_i (J_i Y_i + 1 - J_i)$ and show that A can be expressed as an exponential sum over $F(j, \mathbf{N}, \mathbf{Y})$! This is clearly a $\mathsf{p\text{-}log\text{-}Expsum}$ instance, which finally yields that the τ-complexity of $a(n)$ is subpolynomial. A similar proof strategy also holds for the polynomials with coefficients being definable in $\mathsf{CH_{lin}P}$. For details, see Section 6.

Proof idea of Theorem 1 Take the Pochhammer polynomial $p_n(X) = \prod_{i=1}^{n}(X + i)$. The coefficient of X^{n-k} in p_n will be $\sigma_k(1, \ldots, n)$, where $\sigma_k(z_1, \ldots, z_n)$ is the k-th elementary symmetric polynomial in variables $z_1, \ldots, z_n$. It is not hard to show that $\mathsf{CH_{lin}P}$ is closed under polynomially-many additions and multiplications (Theorem 19; for the proof, see Appendix B). Therefore, $(\sigma_k(1, \ldots, n))_{n\in\mathbb{N}, k\le n}$ is definable in the linear counting hierarchy (see Corollary 20). And by Theorem 21, $(p_n)_{n\in\mathbb{N}}$ has $n^{o(1)}$-sized constant-free circuits if $\mathsf{p\text{-}log\text{-}Expsum}$ is fixed-parameter tractable. But p_n has n distinct integer roots. Assuming the τ-conjecture, $\mathsf{p\text{-}log\text{-}Expsum}$ is not *fpt*. On the other hand, one can show that when exponential sums over circuits of size m have circuits have size $2^{o(n)}\mathsf{poly}(m)$, then the $\mathsf{p\text{-}log\text{-}Expsum}$ is fpt, by Theorem 16; in other words, $\mathsf{p\text{-}log\text{-}Expsum}$ is not fpt implies an exponential lower bound on an exponential sum. This finishes the proof.

Proof idea of Theorem 3. The hardness proof is gadget based (Theorem 41). The details are however quite complicated since we have to cleverly keep track of the cycle lengths. For the upper bound, we work along a tree decomposition. While it is known that the permanent can be computed in fpt time on graphs of bounded treewidth, we cannot simply adapt these algorithms, since we have to produce a formula. This can be achieved using a *balanced tree decomposition*; see Appendix F for definitions and proofs.

1.3 Previous results

To prove (conditional) exponential lower bounds, the standard assumptions that $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{VP} \neq \mathsf{VNP}$ are not enough. It is consistent with our current knowledge that for instance $\mathsf{P} \neq \mathsf{NP}$, but NP-hard problems can have subexponential time algorithms. What we need is a complexity assumption stating that certain problems can only be solved in exponential time. This is the exponential time hypothesis (ETH) in the Boolean setting. Dell et al. [10] studied the exponential time complexity of the permanent, they prove that when there is an algorithm for computing the permanent in time $2^{o(n)}$, then this violates the counting version of the exponential time hypothesis #ETH. #ETH states that there is a constant c such that no deterministic algorithm can count the number of satisfying assignments of a formula in 3-CNF in time 2^{cn}. For connections between parameterized and subexponential complexity in the Boolean setting, we refer to [12, 13].

Bläser and Engels [3] transfer the important definitions and results from parameterized complexity in the Boolean world to define a theory of parameterized algebraic complexity classes. In particular, they define the VW-hierarchy and prove that the clique polynomial and the k-permanent are $\mathsf{VW}[1]$-complete (under so-called fpt-substitutions). They also claim the hardness of the restricted permanent for the class $\mathsf{VW}[t]$ for every constant t and sketch a proof. Note that $\mathsf{VW}[\mathsf{F}]$ contains each $\mathsf{VW}[t]$. So we strengthen the hardness proof in [3] and complement it with an upper bound.

The main tool used by Bürgisser [8] to prove the results above is the counting hierarchy. The polynomial counting hierarchy was introduced by Wagner [28] to classify the complexity of Boolean counting problems. The fact that small circuits for the permanent collapses the counting hierarchy is used by Bürgisser to prove the results mentioned above.

Finally, there have been quite a few works [8, 17, 19, 18], where we have conditional separations on the constant-free version of VP and VNP, namely VP^0 and VNP^0, or their variants, depending on the strength of the conjecture. But this is the first time that we are separating algebraic classes and proving exponential lower bounds, assuming the τ-conjecture.

1.4 Structure of the paper

In Section 2, we defined the basics of constant-free Valiant's model and the unbounded and parameterized setting. In Section 3, we introduce the linear counting hierarchy ($\mathsf{CH}_{\mathrm{lin}}$) and its basic properties. Section 4 connects Valiant's model to the counting hierarchy. Here, we formally introduce exponential sums and investigate their relation to the parameterized classes. The main result is that the fixed-parameter tractability of exponential sums collapses the counting hierarchy. The proofs are quite similar to [8], however, we need to pay special attention to the fact the witness size is linear. Section 5 introduces the definability (computability) of integers in the linear counting hierarchy, and some closure properties of the same. Section 6 proves the exponential lower bound on exponential sum assuming τ-conjecture. Section 7 introduces the parameterized VW-classes and its basic properties. In Section 8 we prove some easy conditional collapse results of the VW-hierarchy in various circuit models. Finally, in Section 9, we prove the completeness of the restricted permanent. Due to space limitations, many proofs and theorems had to be omitted. They can be found in the appendix.

2 Preliminaries I

2.1 Constant-free and unbounded models

Constant-free Valiant's classes: We will say that an algebraic circuit is *constant-free*, if no field elements other than $\{-1,0,1\}$ are used for labeling in the circuit. Clearly, constant-free circuits can *only* compute polynomials in $\mathbb{Z}[\mathbf{X}]$. For $f(X) \in \mathbb{Z}[\mathbf{X}]$, $\tau(f)$ is the size of a minimum size constant-free circuit that computes f, while $L(f)$ denotes the minimum size circuit that computes f. It is noteworthy to observe that, *unlike* Valiant's classical models, computing integers in the constant-free model can be costly; e.g., $\tau(2^{2^n}X^n) = \Omega(n)$, while $L(2^{2^n}X^n) = \Theta(\log n)$. On the other hand, for any $f \in \mathbb{Z}[\mathbf{X}]$, $L(f) \le \tau(f)$.

Before defining the constant-free Valiant classes, we formalize the notion of *formal degree* of a node, denoted formal-deg$(\cdot)$. It is defined recursively as follows: (1) the formal degree of an input gate is 1, (2) if $u = v+w$, then formal-deg$(u) = \max($formal-deg(v), formal-deg$(w))$, and (3) if $u = v \times w$, then formal-deg$(u) =$ formal-deg$(v) +$ formal-deg(w). The formal degree of a circuit is defined as the formal degree of its output node.

The class *constant-free Valiant's* P, denoted by VP^0, contains all p-families (f) in $\mathbb{Z}[\mathbf{X}]$, such that formal-deg(f) and $\tau(f)$ are both p-bounded. Analogously, VNP^0 contains all p-families (f_n), such that there exists a p-bounded function $q(n)$ and $(g_n) \in \mathsf{VP}^0$, where

$$f_n(\mathbf{X}) \;=\; \sum_{\overline{y}\in\{0,1\}^{q(n)}} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)}) \;.$$

It is not clear whether showing $\mathsf{VP}^0 \ne \mathsf{VNP}^0$ implies $\mathsf{VP} \ne \mathsf{VNP}$, it is *not even clear* whether $\mathsf{VP}^0 \ne \mathsf{VNP}^0 \implies \tau(\mathrm{per}_n) = n^{\omega(1)}$. The *subtlety* here is that in the algebraic completeness proof for the permanent, *divisions by two* occur! However, a partial implication is known due to [8, Theorem 2.10]: Showing $\tau(2^{p(n)}f_n) = n^{\omega(1)}$, for some $f_n \in \mathsf{VNP}^0$ and all p-bounded$p(n)$ would imply that $\tau(\mathrm{per}_n) = n^{\omega(1)}$.

Arithmetization is a well-known technique in complexity theory. To arithmetize a Boolean circuit C computing a Boolean function φ, we use the arithmetization technique wherein we map $\varphi(x_1,\ldots,x_n)$ to a polynomial $p(x_1,\ldots,x_n)$ such that for any assignment of Boolean values $v_i \in \{0,1\}$ to the x_i, $\varphi(v_1,\ldots,v_n) = p(v_1,\ldots,v_n)$ holds.

We define the arithmetization map Γ for variables x_i, and clauses $c_1,\ldots,c_m$, as follows:

1. $x_i \mapsto x_i$,
2. $\neg x_i \mapsto 1 - x_i$,
3. $c_1 \vee \cdots \vee c_m \mapsto 1 - \prod_{i\in[m]}(1-\Gamma(c_i))$,
4. $c_1 \wedge \cdots \wedge c_m \mapsto \prod_{i\in[m]}\Gamma(c_i)$.

This map allows us to transform C into an arithmetic circuit for p. For a Boolean circuit C, we denote the arithmetized circuit by $\mathsf{arithmetize}(C)$. Here, we remark that the degree of $\mathsf{arithmetize}(C)$ can become *exponentially* large; this is because there is no known depth-reduction for Boolean circuits, and hence the degree may double at each step, owing to an exponential blowup in the degree.

Valiant's classes in the unbounded setting: It is well-known that an algebraic circuit of size s, can compute polynomials of degree $\exp(s)$; e.g., $f(x) = x^{2^s}$, and $L(f) = O(s)$. This brings us to the next definition, the class $\mathsf{VP}_{\mathrm{nb}}$, originally defined in [20]. A sequence of polynomials $(f) = (f_n)_n \in \mathsf{VP}_{\mathrm{nb}}$, if the number of variables in f_n and $L(f_n)$ are both p-bounded (the degree *may be* exponentially large). The subscript "nb" signifies the *"not bounded"* phenomenon on the degree of the polynomial, in contrast to the original class

VP. Similarly, a sequence of polynomials $(f) = (f_n)_n \in \mathsf{VNP}_{\mathrm{nb}}$, if there exists a p-bounded function $q(n)$ and $g_n(\mathbf{X}, Y_1, \ldots, Y_{q(n)}) \in \mathsf{VP}_{\mathrm{nb}}$ where

$$f_n(\mathbf{X}) = \sum_{\overline{y} \in \{0,1\}^{q(n)}} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)}) .$$

One can analogously define $\mathsf{VP}^0_{\mathrm{nb}}$ and $\mathsf{VNP}^0_{\mathrm{nb}}$, in the constant-free setting. It is obvious that $\mathsf{VP}_{\mathrm{nb}} = \mathsf{VNP}_{\mathrm{nb}}$ implies $\mathsf{VP} = \mathsf{VNP}$, but the converse is *unclear*. However, [20] showed that over a ring of positive characteristic, the converse holds, i.e., $\mathsf{VP} = \mathsf{VNP}$ implies $\mathsf{VP}_{\mathrm{nb}} = \mathsf{VNP}_{\mathrm{nb}}$! On the other hand, [19] showed that $\mathsf{VP}^0 = \mathsf{VNP}^0$ implies that $\mathsf{VP}^0_{\mathrm{nb}} = \mathsf{VNP}^0_{\mathrm{nb}}$, and the converse is unclear because it seems difficult to rule out the possibility that some polynomial family in VNP^0 does not lie in VP^0, but still in VP (i.e., computable by polynomial-size algebraic circuits using *exponentially large-bit* integers).

2.2 Parameterized Valiant's classes

Parameterized Valiant's classes were introduced in [3]. We will briefly review the definitions and results there and extend them to the constant-free and unbounded setting. We first start with the fixed-parameter tractable classes. The W-hierarchies will be introduced later since we only need them in the second part of this work.

Our families of polynomials will now have two indices. They will be of the form $(p_{n,k})$. Here, n is the index of the family and k is the parameter. We will say a polynomial family $(p_{n,k})$ is a *parameterized* p-family if the number of variables is p-bounded in n and the degree is p-bounded in n, k. If there is no bound on the degree, we say it is *parameterized* family.

The most natural parameterization is by the degree: Let (p_n) be any p-family then we get a parameterized family $(p_{n,k})$ by setting $p_{n,k} :=$ the homogeneous part of degree k of p_n. For more details, we will refer the reader to [3].

We now define fixed-parameter variants of Valiant's classes with the constant-free version.

▶ **Definition 4** (Algebraic FPT classes). **1.** *A parameterized p-family $(p_{n,k})$ is in* VFPT *iff $L(p_{n,k})$ is upper bounded by $f(k)q(n)$ for some p-bounded function q and some function $f : \mathbb{N} \to \mathbb{N}$ (such bound will be called an fpt bound). If one removes the requirement of p-family on $p_{n,k}$, and imposes only that the number of variables is p-bounded, one gets the class $\mathsf{VFPT}_{\mathrm{nb}}$.*

2. *A parameterized p-family $p_{n,k}$ is in VFPT^0 iff $\tau(p_{n,k})$ is upper bounded by $f(k)q(n)$ for some constant p-bounded function q and some function $f : \mathbb{N} \to \mathbb{N}$. Similarly, one gets $\mathsf{VFPT}^0_{\mathrm{nb}}$, if one removes the requirement of p-family, and imposes only that the number of variables is p-bounded.*

We remark that in the above, f need not be computable as Valiant's model is non-uniform.

▶ **Definition 5** (Fpt-projection). *A parameterized family $f = (f_{n,k})$ is an fpt-projection of another parameterized family $g = (g_{n,k})$ if there are functions $r, s, t : \mathbb{N} \to \mathbb{N}$ such that r is p-bounded, s, t are computable and $f_{n,k}$ is a projection of $g_{r(n)s(k),k'}$ for some $k' \leq t(k)$,*[2]*. We write $f \leq_p^{\mathrm{fpt}} g$.*

[2] k' might depend on n, but its size is bounded by a function in k. There are examples in the Boolean world, where this dependence on n is used.

However p-projection in Valiant's world seems to be *weaker* compared to parsimonious poly-time reduction in the Boolean world; therefore we need a stronger notion of reduction for defining algebraic models of the Boolean $\#W$-classes, see [3]. That's why we are defining substitutions. We will analogously define it for constant-free model as well.

▶ **Definition 6** (Fpt-substitution). **1.** *A parameterized family $f = (f_{n,k})$ is an fpt-substitution of another parameterized family $g = (g_{n,k})$ if there are functions $r, s, t, u : \mathbb{N} \to \mathbb{N}$ and polynomials $h_1, \dots, h_{u(r(n)s(k))} \in \mathbb{K}[\mathbf{X}]$ with both $L(h_i)$ and $deg(h_i)$ fpt-bounded such that r, u are p-bounded, s, t are computable functions, and $f_{n,k}(\mathbf{X}) = g_{r(n)s(k),k'}(h_1, \dots, h_{u(r(n)s(k))})$ for some $k' \leq t(k)$. We write $f \leq_s^{\mathrm{fpt}} g$. When we allow* **unbounded** *degree substitution of h_i (i.e. only $L(h_i)$ is fpt-bounded), we say that f is an $\mathrm{fpt}_{\mathrm{nb}}$-substitution of g. We denote this as $f \leq_s^{\mathrm{fpt}_{\mathrm{nb}}} g$.*

2. *A parameterized family $f = (f_{n,k})$ is a constant-free fpt-substitution of another parameterized family $g = (g_{n,k})$ if there are functions $r, s, t, u : \mathbb{N} \to \mathbb{N}$ and polynomials $h_1, \dots, h_{u(r(n)s(k))} \in \mathbb{K}[\mathbf{X}]$ with both $\tau(h_i)$ and $deg(h_i)$ are fpt-bounded such that r, u are p-bounded, s, t are computable and $f_{n,k}(\mathbf{X}) = g_{r(n)s(k),k'}(h_1, \dots, h_{u(r(n)s(k))})$ for some $k' \leq t(k)$. We write $f \leq_s^{\tau\text{-fpt}} g$. If we remove the degree condition, we get $\mathrm{fpt}_{\mathrm{nb}}$-substitutions, denoted as $f \leq_s^{\tau\text{-fpt}_{\mathrm{nb}}} g$.*

One can define constant-free fpt-projections analogously. The following lemma should be immediate from the definitions, see [3] for a proof in the case of VFPT.

▶ **Lemma 7.** VFPT, $\mathsf{VFPT}_{\mathrm{nb}}$ *and their constant-free versions* (VFPT^0, $\mathsf{VFPT}^0_{\mathrm{nb}}$) *are* closed *under fpt-projections and fpt-substitutions (constant-free fpt-projections and constant-free fpt-substitutions, respectively).*

3 Linear counting hierarchy

In this section, we define the linear counting hierarchy, a variant of the counting hierarchy, which will allow us to talk about subexponential complexity. The original counting hierarchy was defined by Wagner [28]. We here restrict the witness length to be linear, which is important when dealing with exponential complexity. Allender et al. [2] also define a linear counting hierarchy. Their definition is not comparable to ours. We use an operator-based definition: The base class is deterministic polynomial time and the witness length is linearly bounded. Allender et al. use an oracle TM definition: The oracle Turing machine is probabilistic and linear time bounded, which automatically bounds the query lengths.

▶ **Definition 8.** *Given a complexity class K, we define $\mathbf{C}.K$ to be the class of all languages A such that there is some $B \in K$ and a function $p : \mathbb{N} \to \mathbb{N}$, $p(n) = O(n^c)$ for some constant c, and some polynomial time computable function $f : \{0,1\}^* \to \mathbb{N}$ such that,*

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} : \langle x, y \rangle \in B\}| > f(x).$$

We start from $\mathsf{C}_0\mathsf{P} := \mathsf{P}$ and for all $k \in \mathbb{N}$, $\mathsf{C}_{k+1}\mathsf{P} := \mathbf{C}.\mathsf{C}_k\mathsf{P}$. Then the *counting hierarchy* is defined as $\mathsf{CH} := \bigcup_{k \geq 0} \mathsf{C}_k\mathsf{P}$. We now define our linear counting hierarchy:

▶ **Definition 9.** *Given a complexity class K, we define $\mathbf{C}_{\mathsf{lin}}.K$ to be the class of all languages A such that there is some $B \in K$ and a function $\ell : \mathbb{N} \to \mathbb{N}$, $\ell(n) = O(n)$, and some polynomial time computable function $f : \{0,1\}^* \to \mathbb{N}$ such that,*

$$x \in A \iff |\{y \in \{0,1\}^{\ell(|x|)} : \langle x, y \rangle \in B\}| > f(x).$$

We define $\mathsf{C\text{-}lin}_0\mathsf{P} := \mathsf{P}$ and for all $k \in \mathbb{N}$, $\mathsf{C\text{-}lin}_{k+1}\mathsf{P} := \mathbf{C}_{\mathsf{lin}}.\mathsf{C\text{-}lin}_k\mathsf{P}$. The *linear counting hierarchy* is $\mathsf{CH}_{\mathsf{lin}}\mathsf{P} := \bigcup_{k\geq 0} \mathsf{C\text{-}lin}_k\mathsf{P}$.

Now, we slightly modify the above definition to get $\exists_{\mathsf{lin}}.K$ and $\forall_{\mathsf{lin}}.K$ in the following way: $x \in A \iff \exists y \in \{0,1\}^{\ell(|x|)} : \langle x, y\rangle \in B$ and $x \in A \iff \forall y \in \{0,1\}^{\ell(|x|)} : \langle x, y\rangle \in B$, respectively. Clearly, it can be said that $K \subseteq \exists_{\mathsf{lin}}.K \subseteq \mathbf{C}_{\mathsf{lin}}.K$ and $K \subseteq \forall_{\mathsf{lin}}.K \subseteq \mathbf{C}_{\mathsf{lin}}.K$.

We can define the linear counting hierarchy in a slightly easier manner.

▶ **Definition 10.** *Given a complexity class K, we define $\mathbf{C}'_{\mathsf{lin}}.K$ to be the class of all languages A such that there is some $B \in K$ and a function $\ell : \mathbb{N} \to \mathbb{N}$, $\ell(n) = O(n)$, such that*

$$x \in A \iff |\{y \in \{0,1\}^{\ell(|x|)} : \langle x, y\rangle \in B\}| > 2^{\ell(|x|)-1}$$

It is clear that $\mathbf{C}'_{\mathsf{lin}}.K \subseteq \mathbf{C}_{\mathsf{lin}}.K$ for any class K. Moreover, by an easy adaption of the proof of [25, Lemma 3.3], for any language $K \in \mathsf{CH}$, $\mathbf{C}_{\mathsf{lin}}.K \subseteq \mathbf{C}'_{\mathsf{lin}}.K$. Also, from the definition, we can say that $\mathsf{CH}_{\mathsf{lin}}\mathsf{P} \subseteq \mathsf{CH}$. Therefore, the following holds.

▶ **Fact 11.** $\mathsf{C\text{-}lin}_{k+1}\mathsf{P} = \mathbf{C}'_{\mathsf{lin}}.\mathsf{C\text{-}lin}_k\mathsf{P}$.

We also need a subexponential version of the counting hierarchy. Let $\mathsf{SUBEXP} = \mathsf{DTime}(2^{o(n)})$. Then we set $\mathsf{C\text{-}lin}_0\mathsf{SUBEXP} = \mathsf{SUBEXP}$ and for all $k \in \mathbb{N}$, $\mathsf{C\text{-}lin}_{k+1}\mathsf{SUBEXP} := \mathbf{C}_{\mathsf{lin}}.\mathsf{C\text{-}lin}_k\mathsf{SUBEXP}$. Moreover, $\mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP} = \bigcup_{k\geq 0} \mathsf{C\text{-}lin}_k\mathsf{SUBEXP}$.

Here we define a few more terms that we shall use later in Section 5. We set $\mathsf{NP}_{\mathsf{lin}} = \exists_{\mathsf{lin}}.\mathsf{P}$, NP with linear witness size. In the same way, we can define the levels of the linear polynomial time hierarchy, Σ_i^{lin} and Π_i^{lin}, by applying the operators $\exists_{\mathsf{lin}}$ and $\forall_{\mathsf{lin}}$ in an alternating fashion to P. The linear polynomial hierarchy $\mathsf{PH}_{\mathsf{lin}}$ is the union over all Σ_i^{lin}.

From the above definitions, we get the following conclusion.

▶ **Fact 12.** $\mathsf{NP}_{\mathsf{lin}} \subseteq \mathsf{PH}_{\mathsf{lin}} \subseteq \mathsf{CH}_{\mathsf{lin}}$.

4 Connecting Valiant's model to the counting hierarchy

In this section, we aim to prove that subexponential upper bounds for exponential sums imply a collapse of the linear counting hierarchy (for a definition, see Section 3). To show this, we will define a polynomial family $\mathsf{p\text{-}log\text{-}Expsum}$ and show that $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$ is equivalent to exponential sums having subexponential circuits (Corollary 33). $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$ will imply a collapse of the linear counting hierarchy (Theorem 13).

4.1 log-variate exponential sum polynomial family

In this section, we will define a parameterized log-variate exponential sum polynomial family,

$$\mathsf{p\text{-}log\text{-}Expsum}_{m,k}(g) := \sum_{y \in \{0,1\}^{\ell(n)}} g_n(\mathbf{X}, y) ,$$

where $\mathbf{X}$ has n variables, $\ell(n) = O(n)$, and g_n has circuits of size m ($n = \Omega(\log m)$), and the parameter is $k = \frac{n}{\log m}$. m and k are functions of n. Note that the running parameter of the family is m. When we write $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}$, we mean that $\{\mathsf{p\text{-}log\text{-}Expsum}_{m,k}(g)\}_{m,k} \in \mathsf{VFPT}$ for all families g. We are allowing g to have *unbounded* degree, i.e., g may not necessarily be a p-family. We will also be using constant-free circuits computing g in the constant-free context.

4.2 Collapsing of $\mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP}$

The main theorem of the section is the following:

▶ **Theorem 13.** *If* $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$*, then for every language* L *in* $\mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP}$*, we have a constant-free algebraic circuit* χ_L *so that* $x \in L \implies \chi_L(x) = 1$*,* $x \notin L \implies \chi_L(x) = 0$ *and* χ_L *has size* $2^{o(n)}$*.*

▶ Remark 14. Clearly, $\mathsf{CH}_{\mathsf{lin}}\mathsf{P} \subseteq \mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP}$ and hence, $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$ implies that every language in $\mathsf{CH}_{\mathsf{lin}}\mathsf{P}$ has subexponential-size constant-free algebraic circuits.

▶ **Theorem 15.** *If* $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$*, then* $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ *has circuits of size* $2^{o(n)}\mathsf{poly}(m)$*.*

Proof. Assume that $\mathsf{p\text{-}log\text{-}Expsum}$ has circuits of size $f(n/\log m)\mathsf{poly}(m)$. We can assume that f is an increasing function. Let $i(n) = \max(\{1\} \cup \{j \mid f(j) \le n\})$. $i(n)$ is nondecreasing and unbounded. Moreover, $f(i(n)) \le n$ for all but finitely many n.

We will prove that $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ has circuits of size $2^{n/i(n)}\mathsf{poly}(m)$. If $m \ge 2^{n/i(n)}$, then $f(n/\log m) \le f(i(n)) \le n$, thus there are circuits of size $n \cdot \mathsf{poly}(m) = \mathsf{poly}(m)$. If $m < 2^{n/i(n)}$, then let $\hat{m} = 2^{n/i(n)}$. We can take a circuit C for g and pad it to a circuit $\hat{C}$ of size s with $\hat{m} \le s \le O(\hat{m})$, such that $\hat{C}$ has the same variables as C. Then let $\hat{k} = n/\log \hat{m}$. Thus, $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ has circuits of size $f(\hat{k})\mathsf{poly}(\hat{m}) = n \cdot \mathsf{poly}(2^{n/i(n)})$. ◀

We will need the unbounded version as stated above, but a similar proof also works for the bounded case. The same is true of the non-constant-free version. We will also need the following converse direction:

▶ **Theorem 16.** *Let* $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ *have circuits of size* $2^{o(n)}\mathsf{poly}(m)$ *for each* g *of size* m*. Then* $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$*.*

Proof. Let C_n be a circuit for $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ of size $2^{O(n/i(n))}\mathsf{poly}(m)$ for some nondecreasing and unbounded function i. Let f be a nondecreasing function such that $f(i(n)) \ge 2^n$. We claim that $\mathsf{p\text{-}log\text{-}Expsum}$ has circuits of size $f(k)\mathsf{poly}(m)$ with $k = n/\log m$. If $m \ge 2^{n/i(n)}$, then C_n has size $\mathsf{poly}(m) \le f(k)\mathsf{poly}(m)$. Otherwise, $k = n/\log m \ge i(n)$ and therefore $f(k) \ge 2^n$. Thus, the trivial circuit for $\sum_{y\in\{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$ has size $f(k)\mathsf{poly}(m)$. ◀

5 Integers definable in $\mathsf{CH}_{\mathsf{lin}}\mathsf{P}$

In [8, Section 3], integers are studied that are definable in the counting hierarchy. We adapt this notation to the *linear* counting hierarchy. Formally, we are given a sequence of integers $(a(n,k))_{n\in\mathbb{N},k\le q(n)}$ for some p-bounded function $q : \mathbb{N} \to \mathbb{N}$. We can assume that $|a(n,k)| \le 2^{n^c}$ for some constant c. In other words, the bit-size of $a(n,k)$ is at most *exponential*, as we think n, k has been represented in binary by $O(\log n)$ bits. Now consider two languages,

$$\mathsf{sgn}(a) := \{(n,k) : a(n,k) \ge 0\} \quad \text{and}$$
$$\mathsf{Bit}(|a|) := \{(n,k,j,b) : j\text{th bit of } |a(n,k)| \text{ is } b\}\,.$$

Here in both of these two languages, n, k, j are given in binary representation.

▶ **Definition 17.** *We say an integer sequence* $(a(n,k))_{n\in\mathbb{N},k\le q(n)}$ *for some* p-bounded *function* q *is definable in* $\mathsf{CH}_{\mathsf{lin}}\mathsf{P}$ *whenever both of* $\mathsf{sgn}(a)$ *and* $\mathsf{Bit}(|a|)$ *are in* $\mathsf{CH}_{\mathsf{lin}}\mathsf{P}$*.*

Chinese remainder language: Now, we define another language and make a connection to the definition of an integer sequence to be definable in $\mathsf{CH_{lin}P}$, via the *Chinese remainder representation*. Given that the bit-size of $a(n,k)$ is at most n^c, we consider the set of all primes $p < n^{2c}$. The product of all such primes is $> 2^{n^c}$. Therefore, from $a(n,k) \bmod p$, for all primes $p < n^{2c}$, we can recover $a(n,k)$. Consider

$$\mathsf{CR}(a) \;:=\; \left\{(n,k,p,j,b) : p \text{ prime},\; p \;<\; n^{2c},\; j\text{-th bit of } (a(n,k) \mod p) \text{ is } b\right\} \;.$$

Now we show an essential criterion for a sequence to be in $\mathsf{CH_{lin}P}$. It is an adaption with some additional modifications and observations from [15], which were further implemented in [8, Theorem 3.5].

▶ **Theorem 18.** *Let $(a(n,k))_{n\in\mathbb{N},k\le q(n)}$ be a integer sequence of exponential bit-size ($|a(n,k)| < 2^{n^c}$). Then, $(a(n,k))$ is definable in $\mathsf{CH_{lin}P}$ iff both $\mathsf{sgn}(a)$ and $\mathsf{CR}(a)$ are in $\mathsf{CH_{lin}P}$.*

Now, we can prove an important *closure* property of non-negative integers definable in $\mathsf{CH_{lin}P}$, which we shall use later. For a proof, see Appendix B.

▶ **Theorem 19** (Closure properties)**.** *Let $(a(n,k))_{n\in\mathbb{N},k\le q(n)}$ be a non-negative integer sequence for some p-*bounded *function $q : \mathbb{N} \to \mathbb{N}$ with $a(n,k)$ having bit-size $< n^c$ and it is definable in $\mathsf{CH_{lin}P}$. Consider the sum and product of $a(n,k)$ defined as follows:*

$$b(n) \;:=\; \sum_{k=0}^{q(n)} a(n,k) \quad \text{and} \quad c(n) \;:=\; \prod_{k=0}^{q(n)} a(n,k)\;.$$

Then, both of $(b(n))_{n\in\mathbb{N}}$ and $(c(n))_{n\in\mathbb{N}}$ are definable in $\mathsf{CH_{lin}P}$.

▶ **Corollary 20.** *Take $a(n,k) := \sigma_{n,k}(1,\dots,n)$, $k \le n$, where $\sigma_{n,k}(z_1,\dots,z_n)$ is the k-th elementary symmetric polynomial on variables $z_1,\dots,z_n$. Then, $(a(n,k))_{n\in\mathbb{N},k\le n}$ is definable in $\mathsf{CH_{lin}P}$.*

6 Connecting the counting hierarchy to the τ-conjecture

In this section, we connect the τ-conjecture to the counting hierarchy. Specifically, we show that the collapse of $\mathsf{CH_{lin}P}$ implies that some explicit polynomial, whose coefficients are definable in $\mathsf{CH_{lin}P}$, is "easy". Formally, we prove the following theorem (see Appendix C for the proof):

▶ **Theorem 21.** *Say, $(a(n))_{n\in\mathbb{N}}$ and $(b(n,k))_{k\le q(n),n\in\mathbb{N}}$ are both definable in $\mathsf{CH_{lin}P}$. Here q is some p-*bounded *function. If* p-log-Expsum $\in \mathsf{VFPT}^0_{\mathrm{nb}}$ *then the following holds:*

1. $\tau(a(n)) = n^{o(1)}$,
2. *If $f_n(X) \;:=\; \sum_{k=1}^{q(n)} b(n,k)X^k$ then $\tau(f_n) = n^{o(1)}$.*

▶ **Theorem 22.** *If the τ-conjecture is true, then* p-log-Expsum $\notin \mathsf{VFPT}_{\mathrm{nb}}$.

Proof. Take the Pochhammer polynomial $p_n(X) = \prod_{i=1}^{n}(X+i)$. The coefficient of X^{n-k} in p_n will be $\sigma_k(1,\dots,n)$, where $\sigma_k(z_1,\dots,z_n)$ is the k-th elementary symmetric polynomial in variables $z_1,\dots,z_n$. And $(\sigma_k(1,\dots,n))_{n\in\mathbb{N},k\le n}$ is definable in linear counting hierarchy by Corollary 20. By Theorem 21, $(p_n)_{n\in\mathbb{N}}$ has $n^{o(1)}$ size constant-free circuit if p-log-Expsum is fixed-parameter tractable. But p_n has distinct n many integer roots. So, assuming the τ-conjecture, p-log-Expsum is not *fpt*. ◀

▶ Remark 23. Instead of taking the Pochhammer polynomial, there are many other possible choices for some explicit polynomial, see [8].

Finally, we prove the exponential lower bound for an exponential sum, proving Theorem 1.

▶ **Theorem 24** (Exponential algebraic lower bound). *If the τ-conjecture is true, then there exists an n-variate polynomial family $\sum_{y\in\{0,1\}^n} g_n(X,y)$, which requires $2^{\Omega(n)}$-size circuits.*

Proof. If the τ-conjecture is true, then Theorem 22 shows that p-log-Expsum $\notin \mathsf{VFPT}_{\mathrm{nb}}$. By the contrapositive statement of Theorem 16, the existence of such a hard exponential sum follows. ◀

▶ Remark 25. The family g_n simply is a universal circuit of size polynomial in n, where the polynomial is large enough to simulate the computation of the Turing machine that shows that the n-th Pochhammer polynomial is definable in $\mathsf{CH}_{\mathsf{lin}}\mathsf{P}$.

7 Preliminaries II: The VW-hierarchy

In this section, we define different variants of the VW-hierarchy, which will be analogous to $\#W$-hierarchy, see [3]. We will consider circuits that can have unbounded fanin gates.

▶ **Definition 26** (Weft). *For an algebraic circuit C, the weft of C is the maximum number of unbounded fan-in gates on any path from a leaf to the root.*

For $n \geq k \in \mathbb{N}$, let $\langle {n \atop k} \rangle$ be the set of all vectors in $\{0,1\}^n$ which have exactly k many 1s.

▶ **Definition 27.** **1.** *A parameterized p-family $f_{n,k}(\mathbf{X})$ is in $\mathsf{VW[F]}$ iff there exists a p-bounded function $q(n)$ and p-family $g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ such that* $f_{n,k} \leq_s^{\mathrm{fpt}} \sum_{\overline{y}\in\langle {q(n) \atop k}\rangle} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ *and g_n can be computed by a polynomial-size formula.*

2. *A parameterized family $f_{n,k}(\mathbf{X})$ is in $\mathsf{VW}_{\mathrm{nb}}\mathsf{[F]}$ iff there exists a p-bounded function $q(n)$ and family $g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ such that* $f_{n,k} \leq_s^{\mathrm{fpt}_{\mathrm{nb}}} \sum_{\overline{y}\in\langle {q(n) \atop k}\rangle} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ *and g_n can be computed by a polynomial-size formula.*

3. *A parameterized p-family $f_{n,k}(\mathbf{X})$ is in $\mathsf{VW^0[F]}$ iff there exists a p-bounded function $q(n)$ and p-family $g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ such that* $f_{n,k} \leq_s^{\tau\text{-fpt}} \sum_{\overline{y}\in\langle {q(n) \atop k}\rangle} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ *and g_n can be computed by a constant-free, polynomial-size formula.*

4. *A parameterized family $f_{n,k}(\mathbf{X})$ is in $\mathsf{VW}^0_{\mathrm{nb}}\mathsf{[F]}$ iff there exists a p-bounded function $q(n)$ and family $g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ such that* $f_{n,k} \leq_s^{\tau\text{-fpt}_{\mathrm{nb}}} \sum_{\overline{y}\in\langle {q(n) \atop k}\rangle} g_n(\mathbf{X}, y_1, \ldots, y_{q(n)})$ *and g_n can be computed by a constant-free, polynomial-size formula.*

In some sense, $\mathsf{VW[F]}$ is a substitution of a *weighted sum* of formulas. We will define $\mathsf{VW[P]}$ as a weighted sum as above, but summing over an arbitrary circuit of polynomial-size. Similarly, we can define $\mathsf{VW^0[P]}$, and its counterpart in the unbounded setting, i.e. $\mathsf{VW}_{\mathrm{nb}}\mathsf{[P]}$, and $\mathsf{VW}^0_{\mathrm{nb}}\mathsf{[P]}$.

Finally, we will define the completeness notion:

▶ **Definition 28.** *We will say a parameterized p-family $f_{n,k}$ is $\mathsf{VW[F]}$-hard if every $g_{n,k} \in \mathsf{VW[F]}$, $g_{n,k} \leq_s^{\mathrm{fpt}} f_{n,q}$. Similarly, we can define completeness for $\mathsf{VW[P]}$.*

We can also define completeness and hardness in the constant-free and unbounded models. We will see more about completeness in Section 9.

8 Conditional collapsing of VW-hierarchy and applications

Let us recall the definition of k-degree n-variate ($n \geq k$) *elementary symmetric polynomial* $\sigma_{n,k}(\mathbf{X}) := \sum_{y \in \langle {n \atop k} \rangle} X_1^{y_1} X_2^{y_2} \ldots X_n^{y_n}$. It is known that $(\sigma_{n,k})_n \in \mathsf{VP}^0$, with a simple dynamic programming algorithm; see [16, Section 4]. Let us define a new polynomial family $B_{n,k}(\mathbf{X})$, which will be important in the latter part of the section: $B_{n,k}(\mathbf{X}) := \sum_{t=0}^{n-k} (-1)^t \binom{k+t}{k} \cdot \sigma_{n,k+t}(\mathbf{X})$. The following claim is crucial:

▷ Claim 29. For $y \in \{0,1\}^n$, $B_{n,k}(y) = \begin{cases} 1, & \text{if } y \in \langle {n \atop k} \rangle, \\ 0, & \text{otherwise.} \end{cases}$

Now we are ready to prove the following transfer theorem from the parameterized Valiant's classes to Valiant's algebraic models.

▶ **Theorem 30.** $\mathsf{VW}^0[\mathsf{P}] \neq \mathsf{VFPT}^0 \implies \mathsf{VP}^0 \neq \mathsf{VNP}^0$. *Similarly,* $\mathsf{VW}[\mathsf{P}] \neq \mathsf{VFPT} \implies \mathsf{VP} \neq \mathsf{VNP}$.

Proof. We will prove the contraposition. Assume that $\mathsf{VP}^0 = \mathsf{VNP}^0$. As mentioned before, we know that $(\sigma_{n,k})_n \in \mathsf{VP}^0$. Further, since $k \in [n]$, for $t \leq n-k$, it is trivial to see that $\tau(\binom{k+t}{k}) = n^{O(1)}$. Therefore, for each $0 \leq t \leq n-k$, $(-1)^t \binom{k+t}{k} \cdot \sigma_{n,k+t}(\mathbf{X})$ has a VP^0-circuit. Since VP^0 is closed under polynomially many additions, it follows that $(B_{n,k})_n \in \mathsf{VP}^0$.

Let $q_{n,k} \in \mathsf{VW}^0[\mathsf{P}]$. By definition, there is a polynomial family $p_{n,k}$ of the above form $p_{n,k}(\mathbf{X}) := \sum_{y \in \langle {n \atop k} \rangle} g_n(\mathbf{X}, y)$, where $g_n(\mathbf{X}, \mathbf{Y})$ is in VP^0, such that $q_{n,k} \leq_s^{fpt} p_{n,k}$. By Claim 29, it follows that

$$p_{n,k} = \sum_{y \in \{0,1\}^n} g_n(\mathbf{X}, y) \cdot B_{n,k}(y) \ .$$

We have already proved above that $B_{n,k}$ has $\mathsf{poly}(n)$ sized constant-free circuits. Hence, $g_n(\mathbf{X}, y) B_{n,k}(y)$ has constant-free $\mathsf{poly}(n)$-size circuit. Therefore, by definition and our primary assumption, it follows that $p_{n,k} \in \mathsf{VNP}^0 = \mathsf{VP}^0 \subseteq \mathsf{VFPT}^0$. Since, VFPT^0 is *closed* under constant-free fpt-substitution (Lemma 7), it follows that $q_{n,k} \in \mathsf{VFPT}^0$, implying $\mathsf{VW}^0[\mathsf{P}] \subseteq \mathsf{VFPT}^0$.

The proof in the usual (not constant-free) model also follows essentially along the same line as above. ◀

▶ Remark 31. The above theorem holds in the unbounded regime as well, i.e., $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}] \neq \mathsf{VFPT}^0_{\mathrm{nb}} \implies \mathsf{VP}^0_{\mathrm{nb}} \neq \mathsf{VNP}^0_{\mathrm{nb}}$ (which further implies $\mathsf{VP}^0 \neq \mathsf{VNP}^0$, see [19]). Similarly, $\mathsf{VW}_{\mathrm{nb}}[\mathsf{P}] \neq \mathsf{VFPT}_{\mathrm{nb}} \implies \mathsf{VP}_{\mathrm{nb}} \neq \mathsf{VNP}_{\mathrm{nb}}$.

We now aim to prove a *conditional separation* of $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}]$ and $\mathsf{VFPT}^0_{\mathrm{nb}}$, by showing that $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}] = \mathsf{VFPT}^0_{\mathrm{nb}}$ implies a collapse of the linear counting hierarchy. To show this, we will show that $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}] = \mathsf{VFPT}^0_{\mathrm{nb}} \implies \mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$ (Corollary 33) from which the collapse of the linear counting hierarchy follows.

▶ **Theorem 32.** *Let* $f(\mathbf{X}) = \sum_{y \in \{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$, *where* $\ell(\cdot)$ *is a linear function and* g *is computed by an arithmetic circuit of size* $m = 2^{O(n^c)}$ *for some constant* c. *Then,* $f(\mathbf{X})$ *can be written as*

$$f(\mathbf{X}) = \sum_{e \in \langle {b(m) \atop k} \rangle} G(\mathbf{X}, e) \ ,$$

for some p-$\mathsf{bounded}$ *function* b *and* $k = \ell(n)/\log m$ *and* G *has* $\mathsf{poly}(m)$ *size circuits.*

▶ **Corollary 33.** $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}] = \mathsf{VFPT}^0_{\mathrm{nb}} \implies \mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$.

Proof. In Theorem 32 we have reduced an instance of p-log-Expsum to an instance of $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}]$ with parameter $k = \ell(n)/\log m$. By our assumption $\mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}] = \mathsf{VFPT}^0_{\mathrm{nb}}$ and thus we can say that $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$. ◀

▶ Remark 34. If one restricts p-log-Expsum to exponential sums over g, where g is a p-family (i.e., it has polynomial degree and size), denoted $\mathsf{p\text{-}log\text{-}Expsum}_{bd}$ (bd for bounded-degree), then the above proof similarly implies that $\mathsf{VW}^0[\mathsf{P}] = \mathsf{VFPT}^0 \implies \mathsf{p\text{-}log\text{-}Expsum}_{bd} \in \mathsf{VFPT}^0$.

Similarly, we also prove a lower bound for the class $\mathsf{VW}_{\mathrm{nb}}[\mathsf{P}]$, assuming an fpt lower bound on p-log-Expsum.

▶ **Theorem 35.** *Say that any family* $F_{m,k}(\mathbf{X}) = \sum_{e \in \langle {b(m) \atop k} \rangle} G(\mathbf{X}, e) \in \mathsf{VW}^0_{\mathrm{nb}}[\mathsf{P}]$ *has* $2^{o(n)}\mathsf{poly}(m)$ *size constant-free circuits where* $\tau(G) \le m$, $n := k \log m/c$, *for some constant* c *and* b *is some* p-bounded *function. Then,* $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$.

9 Restricted permanent

A *cycle cover* of a directed graph is a collection of node-disjoint directed cycles such that each node is contained in exactly one cycle. Cycle covers of a directed graph stand in one-to-one relation with permutations of the nodes.

▶ **Definition 36.** *A cycle cover is* (k, c)*-restricted, if it contains one cycle of length* k *and all other cycles have length* $\le c$.

Let $G = (V, E)$ be directed graph and $w : E \to R$ be a weight function. Here R is a ring and typically the ring of polynomials. The weight of a cycle cover C of G is the product of the weights of the edges in it, that is, $w(C) = \prod_{e \in C} w(e)$.

▶ **Definition 37.** *The* (k, c)*-restricted permanent of an edge-weighted directed graph* G *is*

$$\mathrm{per}^{(k,\le c)}(G) = \sum_C w(C),$$

where the sum is over all (k, c)*-restricted cycle covers.*

If $X = (X_{i,j})$ is a variable matrix, then $\mathrm{per}_n(X)$ is the permanent of the complete directed graph with the edge weights $w(i, j) = X_{i,j}$. The (k, c)-restricted permanent family $\mathrm{per}^{(k,\le c)} = (\mathrm{per}_n^{(k,\le c)}(X_n))$, where X_n is an $n \times n$-variables matrix. $\mathrm{per}^{(k,\le c)}$ is a parameterized family, n is the input size, k is the parameter, and c will be some constant to be determined later.

On general graphs, the restricted permanent is very powerful, even if we keep the parameter fixed.

▶ **Proposition 38.** *The* $(2, 2)$*-restricted permanent family is* VNP*-complete.*

If we restrict the underlying graph appropriately, then the restricted permanent is complete for the class VW[F]. Recall that the girth of an undirected graph is the length of a shortest cycle in the graph. When we talk of the girth of a directed graph, we mean the girth of the graph when we disregard the direction of edges. Furthermore, when we talk about the treewidth of a directed graph, we mean the treewidth of the underlying undirected graph. (For the reader's convenience, we recall the definition of treewidth in the appendix, see Definition 43.)

▶ **Definition 39.** *A directed graph $G = (V, E)$ is (c, b)-*nice *if we can partition the nodes $V = V_1 \cup V_2$ into two disjoint sets, such that*

1. *the graph induced by V_1 has girth $> c$ (not counting self-loops),*
2. *every node in V_1 has a self-loop, and*
3. *the graph induced by V_2 has tree-width bounded by b.*
4. *every cycle that contains vertices from V_1 and V_2 has length $> c$.*

Our main result is the following completeness result. The proofs are rather long and can be found in Sections E and F.

▶ **Theorem 40.** *Let c and b be constants. Let (G_n) be a family of (c, b)-nice graphs. Then the (k, c)-restricted permanent is in* $\mathsf{VW[F]}$.

▶ **Theorem 41.** *Let the underlying field have characteristic* 0*. There is a constant b and a family of $(4, b)$-nice graphs (H_n) such that the $(3k, 4)$-restricted permanent of H_n forms a family of* $\mathsf{VW[F]}$*-hard polynomials.*

References

1. Yaroslav Alekseev, Dima Grigoriev, Edward A. Hirsch, and Iddo Tzameret. Semi-algebraic proofs, IPS lower bounds and the τ-conjecture: Can a natural number be negative? *CoRR*, abs/1911.06738, 2019. URL: `http://arxiv.org/abs/1911.06738`, `arXiv:1911.06738`.
2. Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 295–302. IEEE Computer Society, 2001. `doi:10.1109/CCC.2001.933896`.
3. Markus Bläser and Christian Engels. Parameterized Valiant's Classes. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2019/11464`, `doi:10.4230/LIPIcs.IPEC.2019.3`.
4. Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. Algebraic settings for the problem "P≠ NP?". In *The Collected Papers of Stephen Smale: Volume 3*, pages 1540–1559. World Scientific, 2000.
5. Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, 1989.
6. Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. In Zoltán Fülöp and Ferenc Gécseg, editors, *Automata, Languages and Programming, 22nd International Colloquium, ICALP95, Szeged, Hungary, July 10-14, 1995, Proceedings*, volume 944 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1995. `doi:10.1007/3-540-60084-1\_80`.
7. Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Springer, 2000.
8. Peter Bürgisser. On defining integers and proving arithmetic circuit lower bounds. *computational complexity*, 18:81–103, 04 2009. `doi:10.1007/s00037-009-0260-x`.
9. M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.
10. Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. `doi:10.1145/2635812`.

11 Pranjal Dutta. Real τ-conjecture for sum-of-squares: A unified approach to lower bound and derandomization. In *International Computer Science Symposium in Russia*, pages 78–101. Springer, 2021.

12 Jörg Flum and Martin Grohe. Parametrized complexity and subexponential time (column: Computational complexity). *Bull. EATCS*, 84:71–100, 2004.

13 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

14 David G. Glynn. The permanent of a square matrix. *European Journal of Combinatorics*, 31(7):1887–1891, 2010. URL: https://www.sciencedirect.com/science/article/pii/S0195669810000211, doi:https://doi.org/10.1016/j.ejc.2010.01.010.

15 William Hesse, Eric Allender, and D.A. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

16 Stasys Jukna and Georg Schnitger. On the optimality of bellman–ford–moore shortest path algorithm. *Theoretical Computer Science*, 628:101–109, 2016.

17 Pascal Koiran. Valiant's model and the cost of computing integers. *computational complexity*, 13:131–146, 2005.

18 Pascal Koiran. Shallow circuits with high-powered inputs. In *Innovations in Computer Science - ICS*, 2011.

19 Pascal Koiran and Sylvain Perifel. Interpolation in Valiant's theory. *Computational Complexity*, 20:1–20, 2011.

20 Guillaume Malod. The complexity of polynomials and their coefficient functions. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 193–204. IEEE, 2007.

21 Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *J. Complexity*, 24(1):16–38, 2008. doi:10.1016/j.jco.2006.09.006.

22 Herbert John Ryser. *Combinatorial Mathematics*, volume 14 of *Carus Mathematical Monographs*. Mathematical Association of America, 1963.

23 Michael Shub and Steve Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of " NP$\neq$ P?". *Duke Mathematical Journal*, 81(1):47–54, 1995.

24 Sébastien Tavenas. *Bornes inferieures et superieures dans les circuits arithmetiques*. PhD thesis, Ecole Normale Supérieure de Lyon, 2014.

25 Jacobo Torán. Complexity classes defined by counting quantifiers. *J. ACM*, 38:753–774, 07 1991. doi:10.1145/116825.116858.

26 Leslie G. Valiant. Completeness classes in algebra. In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261. ACM, 1979. doi:10.1145/800135.804419.

27 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer Berlin, Heidelberg, 1999. doi:10.1007/978-3-662-03927-4.

28 Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986. doi:10.1007/BF00289117.

A Omitted proofs from Section 4

Proof of Theorem 13. We prove the above statement by induction on the level of $\mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP}$. By definition, $\mathsf{CH}_{\mathsf{lin}}\mathsf{SUBEXP} = \bigcup_{k\geq 0} \mathsf{C}\text{-}\mathsf{lin}_k\mathsf{SUBEXP}$. For $k = 0$, $\mathsf{C}\text{-}\mathsf{lin}_k\mathsf{SUBEXP} = \mathsf{SUBEXP}$. Now by standard arithmetization, we can get a $2^{o(n)}$ size, unbounded degree constant-free circuit for each $L \in \mathsf{SUBEXP}$, so that the above-mentioned condition holds.

Now, by induction hypothesis say, it is true up to k-th level of the hierarchy. We will prove that it is true for the $(k+1)$-th level. Take any $B \in \mathsf{C}\text{-}\mathsf{lin}_{k+1}\mathsf{SUBEXP}$. By Fact 11 and

Definition 10, there exists $A \in \mathsf{C\text{-}lin}_k\mathsf{SUBEXP}$ such that

$$x \in B \iff |\{y \in \{0,1\}^{\ell(|x|)} : \langle x, y\rangle \in A\}| > 2^{\ell(|x|)-1} ,$$

where ℓ is some linear polynomial. By slight abuse of notation, let χ_A denote an algebraic circuit capturing the characteristic function for A, i.e.,

$$\chi_A(x, y) = 1 \iff \langle x, y\rangle \in A .$$

By the induction hypothesis, we can assume that χ_A has size $2^{o(|x|)}$. Now, one can equivalently write the following:

$$x \in B \iff \sum_{y\in\{0,1\}^{\ell(|x|)}} \chi_A(x, y) > 2^{\ell(|x|)-1} .$$

In this way, we get an instance of $\mathsf{p\text{-}log\text{-}Expsum}$, $\sum_{y\in\{0,1\}^{\ell(|x|)}} \chi_A(x, y)$, where the size of χ_A is $m = 2^{o(|x|)}$ and it computes a polynomial of *unbounded degree* (there is no depth-reduction known for Boolean circuits and thus, it cannot be reduced).

As $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathsf{nb}}$, there is an algebraic circuit C such that $C(x) := \sum_{y\in\{0,1\}^{\ell(|x|)}} \chi_A(x, y)$ and C has subexponential-size by Theorem 15.

Trivially, $\tau(2^{\ell(|x|)-1}) \leq \mathsf{poly}(|x|)$. So, we can make C first constant-free and then Boolean by the standard procedure of computing on the binary representation modulo $2^{\ell(n)}$. Let $\tilde{C}$ is the Boolean circuit that computes the highest bit. We just arithmetize $\tilde{C}$ and take $\chi_B = \mathsf{arithmetize}(\tilde{C})$. Each time we convert the arithmetic circuit to a Boolean one and arithmetize the Boolean circuit, we incur only a small polynomial blow-up in size. Therefore, χ_B has subexponential-size, as desired. ◀

B Omitted proofs from Section 5

Proof sketch of Theorem 18. Our argument goes similar to [8, Theorem 3.5], with some further modifications.

At first, let us show that for nonnegative sequences, (a) is definable in $\mathsf{CH_{lin}P} \iff \mathsf{CR}(a) \in \mathsf{CH_{lin}P}$. To show the $\Rightarrow$ direction, start with a Dlogtime-uniform TC^0 circuit family $(\mathcal{C}_n)_n$ which computes the Chinese remainder representation of an n-bit number, modulo primes $p < n^2$, from its binary representation. By [15, Lemma 4.1], $\mathcal{C}_n$ has size $\mathsf{poly}(n)$ and constant depth D. Consider the language

$$L_d := \{(n, k, F, b) : \text{ on input } a(n, k), \text{ gate } F \text{ of } \mathcal{C}_{n^c} \text{ at depth } d \text{ computes bit } b\} ,$$

for $d \in \{0, \dots, D\}$ and (n, k, F) are given by their binary encoding. [8, Theorem 3.5] shows that $L_{d+1} \in \mathbf{C}'.L_d$. But in fact we can say *even stronger* that $L_{d+1} \in \mathbf{C}'_{\mathsf{lin}}.L_d$. This is because when we are given (n, k, F, b) as input by their binary encoding and F is some majority gate at depth $d + 1$, we need to check if $(n, k, G, 1)$ is in L_d for all gates G at depth d connected to F. The lengths of the witnesses $(n, k, G, 1)$ is $O(\log n)$, which is linear in the input. By Dlogtime-uniformity of $(\mathcal{C}_n)_n$, we can check if G is connected to F in polynomial time. And computing the majority of at most $\mathsf{poly}(n)$ many gates can be done by checking

$$|\{G \mid G \text{ connected to } F \text{ and} (n, k, G, 1) \in L_d\}| > 2^{\ell(\log n)-1} ,$$

for some linear function ℓ. Hence, our claim is true. The rest of the proof and the other direction is similar to the argument given in [8]. Hence, (a) is definable in $\mathsf{CH_{lin}P} \iff \mathsf{CR}(a) \in \mathsf{CH_{lin}P}$.

If (a) might have negative entries, then on both sides, we simply add the statement "$\mathsf{sgn}(a) \in \mathsf{CH_{lin}P}$" (on the left hand side implicitly in the definition of definable). ◀

Proof sketch of Theorem 19. The proof is again similar to [8, Theorem 3.10].

Part (i): $(b(n))_{n\in\mathbb{N}} \in \mathsf{CH_{lin}P}$. By [27], we know that iterated addition of n many numbers $0 \leq X_1, \dots, X_n \leq 2^n$, given in their binary representation, can be done by Dlogtime-uniform TC^0 circuits. Say this circuit family is $(\mathcal{C}_n)_n$. $\mathcal{C}_n$ has $\mathsf{poly}(n)$ size and constant depth D. Now, we can take some $\mathcal{C}_{n^{c'}}$ for some suitable constant c' and using the idea same as in Theorem 18, we can say that $(b(n) := \sum_{k=0}^{q(n)} a(n,k))_{n\in\mathbb{N}}$ is definable in $\mathsf{CH_{lin}P}$. Note that while we are summing a polynomial number of numbers, the bit-size for addressing the elements of $a(n,k)$ is $\log n + \log k = O(\log n)$, which is linear in the input size.

Part (ii): $(c(n))_{n\in\mathbb{N}} \in \mathsf{CH_{lin}P}$. We first find a generator of $\mathbb{F}_p^\times$ for a prime p, $p < n^{2c}$. The smallest generator g can be characterized by

$$\forall\, 1 \leq i < p,\ g^i \neq 1 \quad \text{and} \quad \forall\, 1 \leq \hat{g} < g,\ \exists\, 1 \leq j < p,\ \hat{g}^j = 1.$$

The inner checks $g^i \neq 1$ and $\hat{g}^j = 1$ can be done in polynomial time (in $\log n$) by repeated squaring. So checking whether a given g is the smallest generator can be done in (the second level of) $\mathsf{PH_{lin}P}$.

Also, given $u \in \mathbb{F}_p^\times$ and a generator g of $\mathbb{F}_p^\times$, finding $1 \leq i < p$ so that $u = g^i$ can be done in $\exists_{\mathsf{lin}}\mathsf{P}$. Note that,

$$c(n) \mod p = \prod_{k=0}^{q(n)} a(n,k) \mod p = g^{\sum_{k=0}^{q(n)} \alpha(k,n)} .$$

Finding g and $\alpha(n,k)$ is in $\mathsf{CH_{lin}P}$, by the above argument. Moreover, the previous part of the proof also shows that $\sum_{k=0}^{q(n)} \alpha(n,k)$ is definable in $\mathsf{CH_{lin}P}$. Therefore, $(c(n))_{n\in\mathbb{N}}$ is definable in $\mathsf{CH_{lin}P}$. ◀

Proof Sketch of Corollary 20. Consider the polynomial

$$F_n(X) := (X+1)\dots(X+n) = \sum_{k=0}^{n} a(n,k)\cdot X^{n-k} .$$

Substituting X by 2^{n^2}, we get that

$$d(n) := \prod_{j=1}^{n} (2^{n^2}+j) = \sum_{k=0}^{n} a(n,k)\cdot 2^{n^2(n-k)} .$$

And as $a(n,k) < 2^{n^2}$, there is *no overlap* in the bit representations. Hence, it is enough to show that $(d(n))_{n\in\mathbb{N}}$ is definable in $\mathsf{CH_{lin}P}$. And by Theorem 19, we only need to prove that $(e(n,k) := 2^{n^2}+k)_{n\in\mathbb{N},k\leq n}$ is definable in $\mathsf{CH_{lin}P}$, which is indeed true. ◀

C Proofs omitted from Section 6

Proof of Theorem 21. We can assume that if $a(n)$ is definable in $\mathsf{CH_{lin}P}$, $|a(n)| \leq 2^{n^c}$, that is, the bit-size of any integer definable in $\mathsf{CH_{lin}P}$ is polynomially bounded. Furthermore, if $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}}$, then every language in $\mathsf{CH_{lin}P}$ has subexponential-size circuits by Theorem 13. We will use both facts below.

Proof of part (1). Let $a(n) = \sum_{j=1}^{p(n)} a(n,j)2^j$ be the binary decomposition of $a(n)$ and $p(n) = O(n^c)$. Define a new polynomial:

$$A_{\lceil \log n \rceil}(Y_1, \dots, Y_{\mathsf{bit}(n)}) := \sum_{j=0}^{p(n)} a(n,j) Y_1^{j_1} \dots Y_{\mathsf{bit}(n)}^{j_{\mathsf{bit}(n)}} ,$$

where $\mathsf{bit}(n) := \lceil \log(p(n)) \rceil$. By our assumption, we can decide if $a(n,j) = b$ by a subexponential-size circuit, given input n and j in binary. Say, $C_r(\mathbf{N},\mathbf{J})$ is the corresponding circuit, where $r = \lfloor \log n \rfloor$. We have $C_r(n_1,\ldots,n_{\lfloor \log n\rfloor+1}, j_1,\ldots,j_{\mathsf{bit}(n)}) = a(n,j)$, where the n_i's and the j_i's are the bits of n and j, respectively. Consider the polynomial

$$F_r(J_1,\ldots,J_{cr+1},N_1,\ldots,N_{r+1},Y_1,\ldots,Y_{cr+1}) \ := \ C_r(\mathbf{N},\mathbf{J}) \cdot \prod_{i=1}^{cr+1} (J_iY_i + 1 - J_i) \ .$$

Now, by our assumption and Theorem 13, we can say that F_r has $2^{o(r)}$ size constant-free algebraic circuits (of unbounded degree). Consider the exponential sum

$$\tilde{F}_r(\mathbf{N},\mathbf{Y}) \ := \sum_{j\in\{0,1\}^{cr+1}} F_r(j,\mathbf{N},\mathbf{Y}) \ .$$

It is an instance of $\mathsf{p\text{-}log\text{-}Expsum}$ with $\tau(F_r) = 2^{o(r)}$. By assumption, this implies that $\tau(\tilde{F}_r) = 2^{o(r)}$. Finally, note that $A_{\lceil \log n\rceil}(\mathbf{Y}) = \tilde{F}_r(n_1,\ldots,n_{r+1},\mathbf{Y})$, and $a(n) = A_{\lceil \log n \rceil}(2^{2^0},\ldots,2^{2^{\mathsf{bit}(n)-1}})$. Therefore,

$$\tau(a(n)) \ \leq \ \tau(\tilde{F}_r) \ + \ \tau(2^{2^{\mathsf{bit}(n)-1}}) \ \leq \ n^{o(1)} \, ,$$

as desired.

Proof of part (2). Again we can assume that $|b(n,k)|$ has polynomially many bits. Let $b(n,k) = \sum_{j=1}^{p(n)} b(n,k,j)2^j$ be the binary decomposition with $p(n) = O(n^{c'})$ and $q(n) = O(n^c)$. Define

$$B_{\lceil \log n\rceil}(Y_1,\ldots,Y_{\mu(n)},Z_1,\ldots,Z_{\lambda(n)}) \ := \ \sum_{k=0}^{q(n)}\sum_{j=0}^{p(n)} b(n,k,j) Y_1^{j_1}\ldots Y_{\mu(n)}^{j_{\mu(n)}} Z_1^{k_1}\ldots Z_{\lambda(n)}^{k_{\lambda(n)}} \ .$$

Here $\mu(n) := \lceil \log(p(n))\rceil$ and $\lambda(n) := \lceil \log(q(n))\rceil$. Let the variable sets be $\mathbf{J} = (J_1,\ldots,J_{c'r+1}), \mathbf{N} = (N_1,\ldots,N_{r+1}), \mathbf{K} = (K_1,\ldots,K_{cr+1}), \mathbf{Y} = (Y_1,\ldots,Y_{c'r+1}), \mathbf{Z} = (Z_1,\ldots,Z_{cr+1})$, where again $r = \lfloor \log n \rfloor$. Define a new polynomial F_r as follows:

$$F_r(\mathbf{J},\mathbf{K},\mathbf{N},\mathbf{Y},\mathbf{Z}) \ := \ D_r(\mathbf{N},\mathbf{J},\mathbf{K}) \cdot \prod_{m=1}^{c'r+1} (J_mY_m + 1 - J_m) \prod_{s=1}^{cr+1} (K_sZ_s + 1 - Z_s) \ .$$

Like in the previous part of the proof, $(D_r(\mathbf{N},\mathbf{J},\mathbf{K}))_r$ is the circuit family for computing $(b(n,k,j))$. In particular,

$$D_r(n_1,\ldots,n_{r+1},j_1,\ldots,j_{\mu(n)},k_1,\ldots,k_{\lambda(n)}) \ = \ b(n,k,j) \ .$$

By our assumption, D_r has $2^{o(r)}$ size constant-free algebraic circuits (of unbounded degree). Consider,

$$\tilde{F}_r(\mathbf{N},\mathbf{Y},\mathbf{Z}) = \sum_{j\in\{0,1\}^{c'r+1}} \sum_{k\in\{0,1\}^{cr+1}} F_r(j,k,\mathbf{N},\mathbf{Y},\mathbf{Z}).$$

It is an instance of $\mathsf{p\text{-}log\text{-}Expsum}$ with $\tau(F_r)$ is $2^{o(r)}$. Since $\mathsf{p\text{-}log\text{-}Expsum} \in \mathsf{VFPT}^0_{\mathrm{nb}} \implies \tau(\tilde{F}_r) = 2^{o(r)}$. Now, $B_{\lceil \log n \rceil}(\mathbf{Y},\mathbf{Z}) = F_r(n_1,\ldots,n_{r+1},\mathbf{Y},\mathbf{Z})$ and

$$f_n(X) \ = \ B_{\lceil \log n\rceil}(2^{2^0},\ldots,2^{2^{\mu(n)-1}},X^{2^0},\ldots,X^{2^{\lambda(n)-1}}) \ .$$

Therefore, $\tau(f_n) \ \leq \ \tau(B_{\lceil \log n \rceil}) \ + \ \tau(2^{2^{\mu(n)}}) \ + \ \tau(X^{2^{\lambda(n)}}) \ \leq \ n^{o(1)}$, as desired. ◀

D Omitted proofs from Section 8

Proof of Claim 29. For a string $y \in \{0,1\}^n$, we will call the *weight* of y, denoted $\mathrm{wt}(y)$, the number of 1's present in y. Note that if $\mathrm{wt}(y) < k$, then $\sigma_{n,k}(y) = 0$ implying $B_{n,k}(y) = 0$. Similarly if $\mathrm{wt}(y) = k$, then $B_{n,k}(y) = \sigma_{n,k}(y)$, which will be exactly equal to 1. Now if $\mathrm{wt}(y) = k + r$ where $r > 0$, then

$$\begin{aligned} B_{n,k}(y) = \sum_{t=0}^{n-k} (-1)^t \binom{k+t}{k} \cdot \sigma_{n,k+t}(y) &= \sum_{t=0}^{r} (-1)^t \binom{k+t}{k} \cdot \sigma_{n,k+t}(y) \\ &= \sum_{t=0}^{r} (-1)^t \binom{k+t}{k} \cdot \binom{k+r}{k+t} \\ &= \sum_{t=0}^{r} (-1)^t \frac{(k+r)!}{k!t!(r-t)!} \, . \end{aligned}$$

Let us further define the tri-variate polynomial $Q(x,y,z) := (x+y-z)^{k+r} \in \mathbb{Z}[x,y,z]$. Note that the coefficient of x^k in $Q(x,y,z)$ is

$$\sum_{t=0}^{r} y^{r-t} z^t (-1)^t \cdot \frac{(k+r)!}{k!t!(r-t)!} \, .$$

Now putting $y = z = 1$, we get the coefficient exactly equal to $B_{n,k}(y)$; since $r \neq 0$, we can say that the coefficient of x^k in $Q(x,1,1)$ is 0, which finally implies that $B_{n,k}(y) = 0$. ◀

Proof of Theorem 32. Let $f(\mathbf{X})$ be an instance of p-log-Expsum, i.e., $f(\mathbf{X}) = \sum_{y \in \{0,1\}^n} g(\mathbf{X}, y)$, where $g(\mathbf{X}, \mathbf{Y})$ has size m constant-free circuit. Here we mention that, although we just take sum over n *variables* here for the ease of presentation, the same proof also works if we sum over $\ell(n)$ many variables for some linear function ℓ.

Let us partition the variable set $\mathbf{Y} = \{Y_1, \ldots, Y_n\} = E_1 \sqcup \cdots \sqcup E_k$. Here $k = n/\log m$, and for all i, $|E_i| = \log m$. For each $S \subseteq E_i$, we take a *new variable* Z_i^S and we do this for all i. Define $\overline{Z_i} := \{Z_i^S \,:\, S \subseteq E_i\}$ and $\mathbf{Z} = \bigcup_i \overline{Z_i}$. The number of $\mathbf{Z}$-variables is $2^{\log m} \cdot k$, which is polynomial in m.

Let us call an assignment of $\mathbf{Z}$ variables a *good assignment*, if *exactly* one variable in each set $\overline{Z_i}$ is set to be 1. Below we show that there is a one-to-one correspondence between $\{0,1\}$ assignments to the $\mathbf{Y}$ variables and *good assignments* to the $\mathbf{Z}$ variables.

Let φ be a homomorphism from $R[\mathbf{Y}] \to R[\mathbf{Z}]$, where $R := \mathbb{F}[\mathbf{X}]$, such that $\varphi : Y_i \mapsto \prod_{S \subseteq E_i,\, Y_j \notin S} (1 - Z_i^S)$. Let us define $\tilde{g}(\mathbf{X}, \mathbf{Z}) := \varphi(g)$. Now let us fix an assignment $y \in \{0,1\}^n$ to the $\mathbf{Y}$ variables. We construct a corresponding good assignment of $\mathbf{Z}$. For each E_i of $\mathbf{Y}$, we have some $S_i \subseteq E_i$ such that *each* variable of E_i, which is in S_i, gets value 1. The remaining variables in $E_i \setminus S_i$ get value 0 (so that it corresponds to y). Pick this particular $S_i \subseteq E_i$. Note that this S_i is *unique* (it can be the empty set). Now set $Z_i^{S_i} = 1$ and $Z_i^S = 0$, if $S \neq S_i$, for all $i \in [k]$.

Each variable in $\bigcup_i S_i$ gets the value 1 and variables in $\bigcup_i (E_i \setminus S_i)$ are assigned 0. Under the map φ, any $Y_j \in E_1 \setminus S_1$ is replaced by $\prod_{S \subseteq E_1,\, Y_j \notin S} (1 - Z_1^S)$. Since, $S_1 \subseteq E_1$ and $Y_j \notin S_1$, $(1 - Z_1^{S_1})$ occurs in the product. And, hence the product becomes 0. Now, let $Y_\ell \in S_1$ and $\varphi(Y_\ell) = \prod_{S \subseteq E_1,\, Y_\ell \notin S} (1 - Z_1^S)$. As $Y_\ell \in S_1$, $(1 - Z_1^{S_1})$ does not contribute to the product. Thus, under the assignment defined before, $\varphi(Y_\ell)$ becomes 1. This argument holds for any E_i. Therefore, one can conclude that

$$f = \sum_{e:\ e \text{ is a } \textit{good} \text{ assignment}} \tilde{g}(\mathbf{X}, e) \, .$$

Note that the weft of the circuit for $\tilde{g}$ has increased by 1 (from that of g), and the size has also increased by a polynomial (in m) factor. To capture a k-weight good assignment exactly, define a new polynomial $p(\mathbf{Z}) \in \mathbb{F}[\mathbf{Z}]$ as follows:

$$p(\mathbf{Z}) \;:=\; \prod_{i=1}^{k} \left(\sum_{S \subseteq E_i} Z_j^S \right) .$$

Clearly, p has a weft-2 circuit of size $\mathsf{poly}(m)$. Further, it is simple to see that for any k-weight $\{0,1\}$ assignment e to the $\mathbf{Z}$ variables, $p(e) = 1$ iff e is a *good* assignment because from each of the product terms, only one variable will survive. Therefore,

$$f \;=\; \sum_{e \in \langle {b(m) \atop k} \rangle} p(e) \cdot \tilde{g}(\mathbf{X}, e) \;, \qquad \text{where } \; b(m) = |\mathbf{Z}| \;.$$

We set $G(\mathbf{X}, \mathbf{Z}) := p(\mathbf{Z})\tilde{g}(\mathbf{X}, \mathbf{Z})$. By the construction, $\tilde{g}$ has weft $\le t+1$, p has weft ≤ 2, and $\tilde{g}, p$ have $\mathsf{poly}(m)$ size circuits. So, this ends our proof. ◀

▶ Remark 42. The construction above increases the weft by one.

Proof of Theorem 35. Take an instance of $\mathsf{p\text{-}log\text{-}Expsum}$, $f(\mathbf{X}) = \sum_{y \in \{0,1\}^{\ell(n)}} g(\mathbf{X}, y)$, for some $\ell(n) = O(n)$. And g has a constant-free circuit of size m. By Theorem 32, we can make it an instance of $\mathsf{VW}^0[\mathsf{P}]$ and say,

$$f = \sum_{e \in \langle {b(m) \atop k} \rangle} \tilde{g}(\mathbf{X}, e) \;, \qquad \text{where } \; b \; \text{ is } p\text{-bounded}, \;\; k = \ell(n)/\log m$$

By our assumption, f has a constant-free circuit of size $2^{o(n)}\mathsf{poly}(m) = 2^{O(n/i(n))}\mathsf{poly}(m)$ for some unbounded and non-decreasing function $i : \mathbb{N} \to \mathbb{N}$. Let h be a non-decreasing function, so that $h(i(n)) \ge 2^n$. We shall prove that f has $h(k)\mathsf{poly}(m)$ size constant-free circuit. If $m \ge 2^{n/i(n)}$, clearly, f has $\mathsf{poly}(m)$ size constant-free circuit. Otherwise, if $m < 2^{n/i(n)}$, this will imply $i(n) \le n/\log m = k$. And hence, $h(k) \ge 2^n$. So, f has $h(k)\mathsf{poly}(m)$ size constant-free circuit. ◀

E Hardness of the restricted permanent

Before we start with the proof of the $\mathsf{VW}[\mathsf{F}]$-hardness, we first give the omitted proof of Proposition 38:

Proof of Proposition 38. We reduce from the matching polynomial on undirected graphs. Given a matching M of the complete undirected graph, we can map it to a $(2,2)$-restricted cycle cover C of the complete directed graph (with self-loops), by mapping each edge $\{i,j\} \in M$ to the 2-cycle $(i,j),(j,i)$. Nodes that are not covered by M are covered by self-loops in C. This is a one-to-one correspondence. Therefore, if we substitute $X_{i,i} = 1$, $1 \le i \le n$ and $X_{i,j} = 1$ for $i > j$, then we get the matching polynomial out of $\mathrm{per}_n^{(2,\le 2)}(X)$. ◀

For the reader's convenience, we recall the definition of tree-width:

▶ **Definition 43.** **1.** *A* tree decomposition *of an undirected graph $G = (V,E)$ is a pair $(\{X_i \mid i \in I\}, T = (I,F))$, where $\{X_i \mid i \in I\}$ is a collection of subsets of V and $T = (I,F)$ is a tree such that:*

- $\bigcup_{i \in I} X_i = V$.

- *For all $\{v, w\} \in E$, there exists an $i \in I$ with $v, w \in X_i$.*
- *For every $v \in V$, $T_v = \{i \in I \mid v \in X_i\}$ is connected in T.*

2. *The* width *of a tree decomposition is* $\max_{i \in I} |X_i| - 1$. *The* treewidth *of a graph G is the minimum width over all the tree decompositions of G.*

The X_i are also called *bags*. The treewidth of a directed graph is the treewidth of the underlying undirected graph.

Now, we are given a formula F in variables $X_1, \dots, X_m$ and $Y_1, \dots, Y_n$. We call the polynomial computed by F also $F(X_1, \dots, X_m, Y_1, \dots, Y_n)$. We are interested in the polynomial

$$P(X_1, \dots, X_m) = \sum_{e \in \left\langle {n \atop k} \right\rangle} F(X, e)$$

We assume that the formula is layered, that is, along each path the addition and multiplication gates alternate. The top gate is an addition gate and each input gate is fed into a multiplication gate.

Our goal is to write P as an fpt-projection of a (k, c)-restricted permanent on a (c, b)-nice graph H for certain constants c and b. The construction will have two main components. One corresponds to the formula F, the other one is similar to the rosetta graph in Valiant's proof of the #P-hardness of the permanent, see e.g. [7]. The first component will be the bounded treewidth part of H, the second one will be the high girth part.

E.1 The graph G_1

We first design a graph G_1 from F. G_1 will have *iff-coupled* edges (pairs of edges). When we have a pair of iff-coupled edges, then either both of them appear in a cycle cover or none of them, see also [7]. Later, we will enforce this by connecting iff-coupled edges with appropriate gadgets.

A cycle cover of G_1 is *consistent* if it contains either both edges of such a pair or none. G_1 will have the property that

- the sum of the weights of all consistent cycle covers in G_1 is F
- and each cycle cover has cycles of length at most two.

The graph will be constructed in an iterative manner, adding new nodes and edges gradually. Each parse tree of F will correspond to one consistent cycle cover and vice versa. Recall that a *parse tree* of an algebraic formula is a subtree that contains the root, for each multiplication gate it contains all children and for each addition gate it contains exactly one child, see [21].

E.1.1 Input gates

Input gates are realized as depicted in Figure 1. Assume that L is the label of the input gate. The gate has the following properties:

- If the top node is externally covered (meaning that the gate is in the parse tree), then there is exactly one consistent cycle cover with weight L (middle, drawn in blue).
- If the top node is uncovered, then there is exactly one consistent cycle cover with weight 1 (right-hand side, drawn in blue).

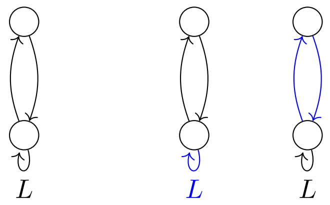


Figure 1 The input gadget and the two ways how to cover it (drawn blue).

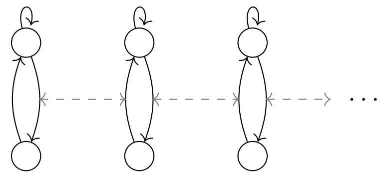


Figure 2 The multiplication gadget. Iff-couplings are drawn as dashed bidirected edges.

E.1.2 Multiplication gates

Multiplication gates are realized as depicted in Figure 2. For each child of the multiplication gate, we have one 2-cycle. These 2-cycles are iff-coupled. The bottom node of each 2-cycle will be the top node of an input gate or the yet-to-define addition gate.

The gate has the following properties:

- If the left-most edge is in a consistent cycle cover, then this consistent cycle cover contains all 2-cycles of the gadget. (This means that the multiplication gate is in the parse tree.)
- If the left-most edge is not in a consistent cycle cover, then all two nodes will be covered by self-loops. The bottom nodes have to be covered externally. (This means that the multiplication gate is not in the parse tree.)

The bottom nodes of each 2-cycle will be the top-node of the input gates or (yet to be defined) addition gates that are fed into the multiplication gate.

- If the multiplication gate is in the parse tree, then all its children are in the parse tree. In this case, the bottom nodes of the 2-cycles are covered within the multiplication gadget. These bottom nodes are the top nodes of the input gadgets and addition gadgets. For these gates, their top nodes are now externally covered, which means that the corresponding gates are in the parse tree, as it should be.
- If the multiplication gate is not in the parse tree, then all its children are not in the parse tree. In this case, the bottom nodes are not covered within the multiplication gadget. Hence, they need to be covered in the input gadgets or additions gates, which means that the corresponding gates are not in the parse tree, too, see the following subsections.

E.1.3 Addition gates

The addition looks as drawn in Figure 3. It has a 2-cycle at the top and then has one 2-cycle for each child. It has the following properties:

- If the top node is not covered externally (that is, the addition gate is not in the parse tree), then there is exactly one consistent cycle cover.
- If the top node is covered externally (that is, the gate is in the parse tree), then there are t different cycle covers, where t is the number of children, one for each 2-cycle in the bottom row. This reflects the fact that in a parse tree, an addition gate has exactly one child.

The Figure 4 shows the situation when the top node is not covered externally on the left-hand side. On the right-hand side, it shows the situation in the second case. Here are t covers, each of them contains one 2-cycle and $t-1$ self-loops.

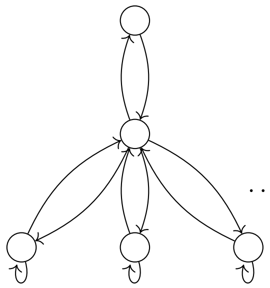


Figure 3 The addition gadget. If the corresponding gate has fanin t, then there are t nodes at the bottom.

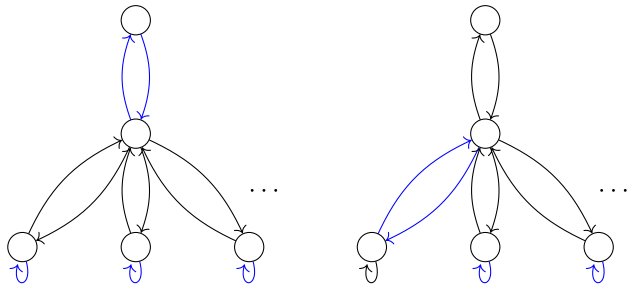


Figure 4 Lefthand side: The covering (drawn blue) if the top node is not externally covered. Righthand side: The covering if the top node is externally covered. One of the bottome nodes is covered by a 2-cycle. This is the child in the corresponding parse tree.

The children of an addition gate are all multiplication gates. The left-most edge of the multiplication gate will be iff-coupled to one of the edges of the corresponding 2-cycle in the bottom row.

E.1.4 Putting it all together

Let F be the given formula. We construct the corresponding graph G_F recursively:

- If F consists of one node (an input node), then G_F is the corresponding input gadget.
- If the top gate of F is a multiplication gate, then let $F_1, \ldots, F_t$ be its children (summation gates). We take the graphs $G_{F_1}, \ldots, G_{F_t}$ and identify their top nodes with the bottom nodes of the corresponding 2-cycles in the multiplication gadget to get G_F.
- If the top gate of F is an addition gate with children $F_1, \ldots, F_t$, then we take the corresponding graphs $G_{F_1}, \ldots, G_{F_t}$ and iff-couple the left of the left-most 2-cycle of the top addition gate to one of the edges of the corresponding 2-cycle of the addition gate.

The graph G_1 will now be the graph G_F with one 2-cycle attached to the top node, when the top node is an addition gate. This ensures that the top node of the addition gadget is always externally covered, so the addition gate is always in a parse tree.

Using induction, we can prove:

▶ **Lemma 44.** *There is a one-to-one correspondence between parse trees P of F and consistent cycle covers C of G_1. The monomial of P equals the weight of F. Furthermore, all cycles in a consistent cycle cover of G_1 have length at most two.*

Proof. For a subformula H of F, G_H denotes the graph defined at the beginning of Section E.1.4 We prove the following more general statement:

- There is a one-to-one correspondence between parse trees P of H and consistent cycle covers C of G_H not covering the top node (in the case of addition and input gates) or not containing the self-loop at the top of the first 2-cycle (in the case of multiplication gates, respectively).
- The monomial of P equals the weight of C.
- If H is an input gate, then here is exactly one cycle cover of G_H where the top node is not covered externally. This cycle cover has weight 1.
- If the top gate of H is an addition gate, then there is exactly one cycle cover of G_H where the top node is not covered externally. This cycle cover has weight 1.

- If the top gate of H is a multiplication gate, then there is exactly one cycle cover of G_H where the top node of the first 2-cycle is covered by the self-loop. This cycle cover has weight 1.
- All cycles in a consistent cycle cover of G_H have length at most two.

The proof is by structural induction. If H consists of one node, then it is an input gate. Let L be its label. H has one parse tree with label L. On the other hand, there is exactly one consistent cycle cover not covering the node at the top. The weight of this cover is L. If the top node is covered, then C consists of the 2-cycle like in Figure 1 on the right-hand side. Its weight is 1.

If the top gate of H is an addition gate, then let $H_1, \dots, F_t$ be its children, which have a multiplication gate at the top. If the top node of the top addition gadget of G_H is not covered, then there are ℓ ways to cover the addition gate, as depicted on the right-hand side of Figure 4. Each parse tree of H is a parse tree of some H_τ, $1 \leq \tau \leq t$, plus one additional edge. By the induction hypothesis, there is a one-to-one correspondence between parse trees of H_τ and consistent cycle covers of G_{H_τ} not containing the self-loop at the top of the first 2-cycle. (Since the formula is layered, the top gates of $H_1, \dots, H_t$ are multiplication gates. Hence, there is also a one-to-one correspondence between cycle covers of G_H and parse trees of H, since by the induction hypothesis, there is only one cover for the subgraphs corresponding to $H_{\tau'}$, $\tau' \neq \tau$, and they all have weight 1. Thus, the weight of the cover of H equals the weight of the cover of H_τ. If the top node of the addition gadget of G_H is covered, then we are in the situation of the left-hand side of Figure 4. By the induction hypothesis, there is only one way to cover each of the subgraphs G_{H_τ}, too. The total weight of this cover is 1.

Finally, if the top gate of H is a multiplication gate with subformalas $H_!, \dots, H_t$, then every parse tree of H consists of parse trees of $H_1, \dots, H_\ell$. If the first 2-cycle is covered, then all 2-cycles are covered. Therefore, the top nodes of G_{H_τ}, $1 \leq \tau \leq t$, are all covered externally, and since the top gates of the H_τ are either addition or input gates, there is one cover of each parse tree of H_τ and this cover has weight equal to the corresponding monomial. The weight of the corresponding cover of G_H is the product of these weights/monomials, and therefore, the weight equals the monomial of the parse tree. If the first 2-cycle is not covered, then none of the 2-cycle is covered. Therefore, the subgraphs G_{H_τ} have only one cover and this cover has weight 1.

The fact that no cover has cycle of length > 2 follows from the fact that no gadget has cycles of length > 2. ◀

E.2 The enumeration gadget

We are given a formula $F(X_1, \dots, X_m, Y_1, \dots, Y_n)$ and we want to sum over the Y-variables. We will represent each Y_i by a directed edge $y_i = (s_i, t_i)$. These edges will be called *y-edges.* There will be directed edges from each t_j to each s_ℓ except for $j = \ell$ connecting the y-edges. These edges will be called *connecting edges.* Each s_ℓ and t_j gets a self-loop. Call the resulting graph R_n. The graph R_n has the following properties:

- Each directed cycle that is not a self-loop has even length, every second edge if a y-edge and every other edge is a connecting edge.

▶ **Lemma 45.** 1. *For every set of k y-edges, there are $k!$ many $(2k, 1)$-restricted cycle covers containing these y-edges and no other y-edges.*

2. *Every cycle cover that is $(2k, c)$-restricted and contains more than k y-edges fulfills $c \geq 4$.*

Proof. The y-edges can be visited in any order. Any two y-edges can be connected by a unique connecting edge. Thus there are $k!$ cycles of length $2k$ covering a given set of y-edges of size k. The remaining nodes can be covered by self-loops.

A cycle of length $2k$ has exactly k y-edges. Thus to cover more than k y-edges, we need a second cycle. Except for the self-loops, the shortest cycles in R_n have length four. ◀

E.3 The graph G_2

The graph G_2 is built as follows.

- We take the graph G_1.
- We add an enumeration gadget R_n.
- Let $\ell_1, \dots, \ell_s$ be the loops of the input gadgets that are labeled with Y_i. We iff-couple the y_i-edge of R_n with ℓ_1, ℓ_1 with ℓ_2, and so on. We do so for every $1 \leq i \leq n$.
- We replace all the weights Y_i by 1.
- Furthermore, we add a self-loop to the top-node of every input gate that was labeled with Y_i. This gives two ways to cover such a gadget when it is not in a parse-tree. One as before with a 2-cycle and the other one with two self-loops. This will be important, since selecting the loop that corresponds to Y_i means setting it to 1, independent of whether it is in a parse-tree or not. However, only one of the two local covers can be chosen, depending on whether Y_i is set to 1 or not.
- If Y_i is set to 0 and the corresponding input gate is in the parse tree, then there is no consistent cycle cover anymore. This is all right, since the corresponding monomial contains Y_i, which is set to 0. If the input gate is not in the parse tree, then is can locally be covered by the 2-cycle.

▶ **Lemma 46.** *The cycle of length $2k$ in every consistent $(2k, 2)$-cycle cover of G_2 is contained in R_n.*

Proof. The longest cycle in G_1 has length two. Thus, the cycle of length $2k$ can only be in R_n ◀

A consistent $(2k, 2)$-restricted cycle cover cannot have any other cycles with y-edges in R_n. We call two consistent $(2k, 2)$-restricted cycle covers y-equivalent if they contain the same y-edges.

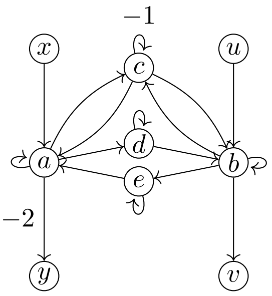
Let $F(X, Y)$ be our given formula. For a $\{0, 1\}$-assignment η to the Y-variables, let F_η denote the resulting formula.

▶ **Lemma 47.** *There is a bijection of parse trees of F_η with nonzero monomial M and the equivalence classes of the $(2k, 2)$-restricted cycle covers with nonzero weight.*

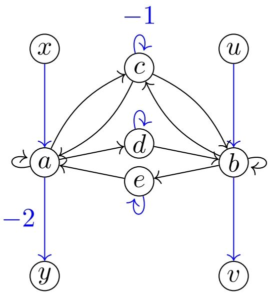
Proof. There is a one-to-one correspondence between the parse trees of F and the consistent cycle covers of G_1. If a parse tree P has a nonzero monomial in F_η, then in F, the monomial can only contain Y_i-variables, that are set to 1 under η. ◀

E.4 The graph G_3

Now the graph G_3 is obtained by replacing the iff-couplings with the gadget in Figure 5. If the two edge (x, y) and (u, v) are iff-coupled, then we subdivide the edges with the nodes a and b and connect them with the subgraph as depicted. For each iff-coupling, we insert a new subgraph. If we do not write a weight explicitly, then the weight of the edge is 1. Similar gadgets were developed in the past, see e.g. [10]. The difference in our gadget is that we have



■ **Figure 5** The iff-gadget. The edges (x, y) and (u, v) are the iff-coupled edges in the original graph.



■ **Figure 6** The covering of the iff-gadget if both edges (x, y) and (u, v) appear in the original cycle cover.

a 4-cycle between a and b instead of a 2-cycle. This will be crucial, since (k, c)-restricted cycle covers are sensitive to changes of cycle lengths.

E.4.1 Local coverings of the iff-gadget

There are essentially four different cases how an iff-gadget can be covered:

- If both edges are taken in G_2, then there is one way to cover the iff-coupling internally, drawn in blue in Figure 6. The contribution to the overall weight of a cover is $(-2) \cdot (-1) = 2$.
- If both edges are not taken, then there are six ways how to cover the gadget locally, shown in Figure 7. Two of them have weight -1, four have weight 1. The overall contribution to the weight is 2.
- If one edge is taken but the other one is not, then there are two ways to cover the gadget. These covers have opposite sign. See Figure 8. The situation when the other edge is taken is symmetric.
- Then there is finally the case when the gadget is covered inconsistently, that is, it is entered via x and left via v. Again there are two covers with opposite signs, see Figure 9. Again, there is a symmetric case.

E.4.2 Consistent cycle covers

A consistent cycle cover C of G_2 are mapped to cycle covers of G_3 where each iff-gadget is covered consistently. If both edges of an iff-gadget are taken, then there is one way to cover the gadget internally. This gives a multiplicative factor of 2. If both edges are not taken, then there are six ways to cover the gadget internally. Again, the overall contribution is 2. If there are M gadgets in total, then C will get mapped to a bunch of cycle cover in this way with total weight $2^M w(C)$.

If the cycle cover C is $(2k, 2)$-restricted, the the resulting cycle covers will be $(3k, 4)$-restricted, since each iff-coupled edge is subdivided. Each edge is only subdivided once except for the loops at the input gates that were labeled with a Y-variable. These are subdivided twice, yielding a cycle of length 3. Furthermore, the internal cycles of the iff-gadgets have length at most 4.

On the other hand, if there is a $(3k, 4)$-restricted cycle cover such that all iff-gadget are covered consistently, then this corresponds to exactly one $(2k, 2)$-restricted consistent cycle cover of G_2. The cycle of length $3k$ will be contained in the R_n-part of G_3.

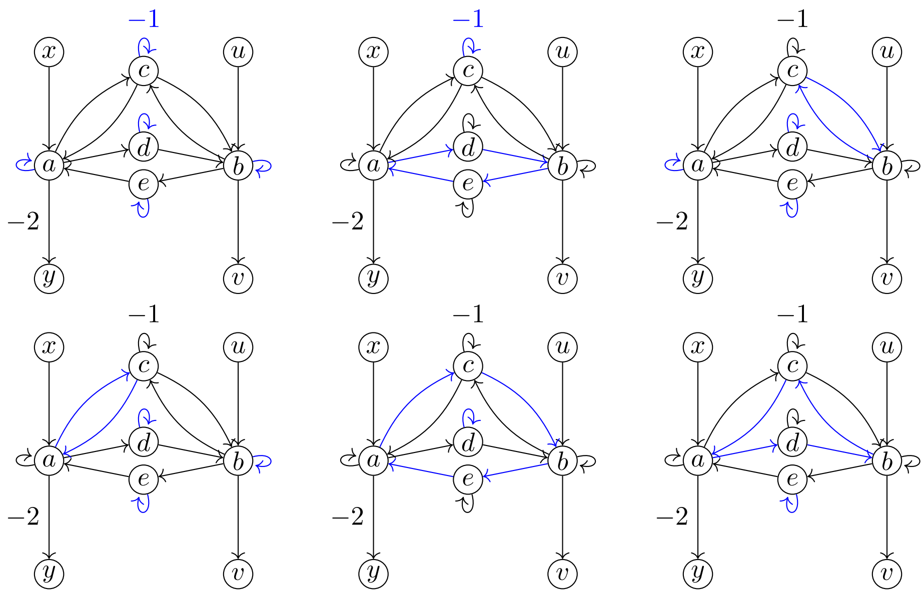


Figure 7 The six ways to cover an iff-gadget consistently, when both edges (x, y) and (u, v) are not in the original cycle cover.

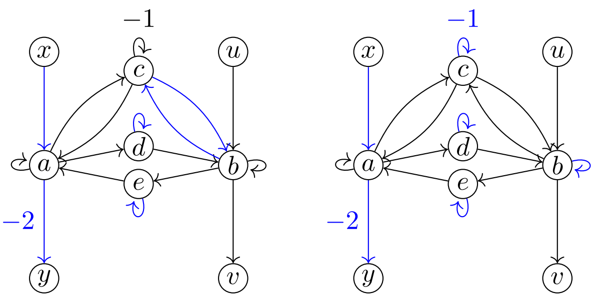


Figure 8 The two ways to cover an iff-gadget if one edge (x, y) is in the original cover and the other one (u, v) is not. Both covers have opposite signs.

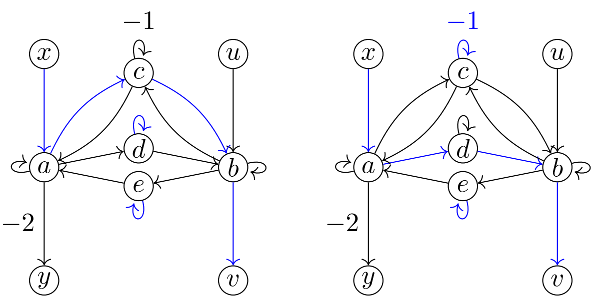


Figure 9 The two ways to cover an iff-gadget if the gadget is entered on the one side and left on the other.

E.4.3 Inconsistent cycle covers

To get rid of the inconsistent cycle covers, we define an involution on the set of inconsistent cycle covers. A cycle cover is inconsistent if at least one iff-gadget is not covered consistently. We define an involution on the set of all inconsistent cycle covers as follows: We number the iff-gadgets arbitrarily. Let C be an inconsistent cycle cover and let I be the first iff gadget that is not covered consistently. We map C to the cycle cover C' where I is covered in the other way as depicted in the Figures 8 and 9. This new cycle cover has weight $w(C') = -w(C)$. The mapping $C \mapsto C'$ is an involution by construction. Finally, if C is (k,c)-restricted for some $c \geq 2$, then C' is also (k,c)-restricted. This is obvious in the first case, since here only the local covering is changed. In the second case, the length of the path that crosses I is not changed (this is the important change to the gadget!), and therefore all cycle-length stay the same. Thus, the overall contribution of the inconsistent cycle covers sums up to 0.

Altogether, this proves the main result of the present section.

▶ **Theorem 48.** *Let F be a layered formula in variables $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$. Let G_3 be the graph defined as above. Then*

$$\mathrm{per}^{(3k,\leq 4)}(G_3) = k! \cdot 2^M \cdot \sum_{e \in \langle{n \atop k}\rangle} F(X,e)$$

where M is the number of iff-couplings

▶ Remark 49. The factor $k!$ comes from the number of $3k$-cycles in the R_n-part. This can be avoided by letting 'all connecting edges (t_i, s_j) with $i > j$ go through the same new node b. In this way, the y-edges have to be visited in ascending order. The factor 2^M seems to be unavoidable though.

Proof of Theorem 41. Let F_n be a universal formula that can simulate any formula of size $\leq n$. Consider the graph G_1 (see Section E.1) and replace the iff-couplings of the multiplication gadgets by an edge, that subdivides and connects the iff-coupled edges of G_1. The resulting graph is essentially a tree that has some 2-cycles and 4-cycles. It is easy to see that the treewidth of this graph is 2. Now in G_3, instead of the edges between iff-coupled edges, we have the iff-gadget. They introduce 3 more nodes, therefore, the treewidth of this part of G_3 is bounded by 5. The other part of G_3 has girth > 4.

Thus, the family H_n will be the graphs G_3 corresponding to F_n. By our construction, every family in VW[F] is reducible to this family. ◀

F Upper bound

▶ **Lemma 50.** *Let $G = (V_1 \cup V_2, E)$ be a (c,b)-nice graph and C be a (k,c)-restricted cycle cover. Let c be the cycle of length k in C. Then all nodes of V_1 that are not in c are covered by self-loops in C.*

Proof. Since $G[V_1]$ has girth $> c$ (except for self-loops), the only cycles of length $\leq c$ in $G[V_1]$ are self-loops. ◀

Let G be an arbitrary edge-weighted graph. We define $\mathrm{per}^{(\leq c)}(G) = \sum_C w(C)$ where the sum is taken over all cycle covers with all cycles having length $\leq c$.

▶ **Theorem 51.** *Let G be a graph of bounded tree-width. Then there is an algebraic circuit of fpt size that computes $\mathrm{per}^{(\leq c)}(G)$.*

Proof. Bodlaender and Hagerup [6] show that whenever a graph has bounded tree-width, then there is a binary tree-decomposition of logarithmic height. Moreover, we can assume that the tree decomposition is nice, see e.g. [9] for a definition, and the height is still logarithmic.

Let $T = (I, F)$ be a nice tree decomposition of G of logarithmic height. For each node $i \in I$, let V_i be the set of all nodes that appear in the subtree below i but not in X_i.

A *path-cycle cover* of a graph is a collection of node disjoint path and cycles. Following the tree-decomposition, we will construct inductively path-cycle covers. Eventually, all paths need to be closed to a cycle in the computation of $\mathrm{per}^{(\le c)}(G)$.

For each node i, we construct circuits computing certain polynomials $P_{i,C}$ with C being a path-cycle cover containing all nodes of X_i and potentially some nodes from V_i, however each path or cycle has to contain at least one node of X_i. Each path has length $\le c - 2$ and each cycle has length $\le c$. The cover C contains a constant number of nodes, since the path and cycles have length bounded by a constant.[3] In the cover, we treat uncovered nodes as path of length 0. We construct the polynomials inductively:

- If the node i is a leaf, then X_i is empty and there is only one polynomial $P_{i,\emptyset} = 1$.
- If i is an introduce node, let x the introduced node. Let $P_{j,D}$ be a polynomial computed at the (unique) child j of i. For each such polynomial, there might be several ways how x can be added to the cover D yielding a new cover C. Each such new cover C gives a polynomial $P_{i,C}$:
  - $P_{i,C} = P_{j,D}$, where C is obtained from D by adding the path x of length 0.
  - $P_{i,C} = w(x, u) \cdot P_{j,D}$ for each u such that there is a path p starting in u of length $\le c - 3$ and there is an edge (x, u). The cover C is obtained from D by prepending x to p.
  - $P_{i,C} = w(v, x) \cdot P_{j,D}$ for each v such that there is a path p ending in v of length $\le c - 3$ and there is an edge (x, u). The cover C is obtained from D by appending x to p.
  - $P_{i,C} = w(v, x)w(x, u')P_{j,D}$ for all paths p ending in v and paths p' starting in u' such that there are edges (v, x) and (x, u') and the total length of the resulting path is $\le c - 2$. C is obtained from D by connecting p and q using x.
  - $P_{i,C} = w(v, x)w(x, u)P_{j,D}$ for each path p from u to v of lenght $\le c - 2$ such that there are edges (v, x) and (x, u). C is obtained from D by closing the path p using x.
  - $P_{i,C} = w(x, x) \cdot P_{j,D}$ if (x, x) is an edge, C is obtained from D by adding a self-loop.
- If i is a forget node, then $P_{i,C} = P_{j,D}$ if x is covered by a cycle c in D or x is not the start or end node of a path p. If there are not any nodes of c still in X_i, then we remove c from D to obtain C. Otherwise, $C = D$. If x is the start or end node of a path, then we simply drop $P_{j,D}$, since we cannot cover x by a cycle after it is forgotten.
- If i is a join node, let $P_{j,D}$ and $P_{j',D'}$ denote the polynomials computed at the two children j and j' of i. For a given cover C at i, we have

  $$P_{i,C} = \sum_{D,D'} P_{j,D} \cdot P_{j',D'}$$

  where D and D' run over all covers such that all cycles and all path of length > 0 that only contain nodes of X_i appear in D, all cycles that contain nodes of V_j appear in D and all cycles that contain nodes of $V_{j'}$ appear in D'. Note that path and cycles that

[3] Note, however, that the bound on the treewidth is what matters. If the bound c was not constant, then the problem would still be fpt. However, the description of the construction would be more complicated.

only contain nodes of X_i could also appear in D'; by forcing them to appear in D, we make the decomposition of C into D and D' unique.

- Finally, if i is the root, then $X_i = \emptyset$. The child j of i is a forget node. We set $P_{i,\emptyset} = P_{j,\emptyset}$.

From the construction it is clear that the polynomial computed at the root is the restricted permanent $\mathrm{per}^{(\leq c)}(G)$. By following the tree decomposition from the leaves to the root, we get an algebraic circuit of fpt size. ◀

▶ Remark 52. We can expand the algebraic circuit constructed in Theorem 51 into a formula. Since the tree decomposition has only logarithmic height, the size of the formula will be $f(b)^{O(\log n)} = n^{O(\log f(b))}$ where b is the treewidth and $f(b)$ is some function of b.

Proof of Theorem 40. Let $G_n = (V, E)$, $|V| = n$ with a partition of $V = V_1 \cup V_2$ as in Definition 39, $n_i = |V_i|$. Consider a (k, c)-restricted cycle cover C of G_n. Let c_1 be the cycle of length k in C. Then by Lemma 50 all nodes in V_1 that are not covered by c_1 are self-loops. This suggests the following approach. We enumerate all sets of size k, check whether they form a cycle. If yes, we cover the remaining nodes in V_1 by self-loops. The remaining nodes induce a graph of bounded tree-width, and we can use Theorem 51 and even Remark 52.

We have variables $E_{i,j}$, $1 \leq i, j \leq n$ representing the edges of the graph. We select k of them using the bounded summation representing the cycle of length k. We first construct a polynomial $\mathrm{Cyc}(E)$ such that $\mathrm{Cyc}(e) = 1$ if $e \in \{0,1\}^{n \times n}$ is the adjacency matrix of a k-cycle and $\mathrm{Cyc}(e) = 0$ otherwise. Since there is a Boolean formula of polynomial-size which checks this, we get an algebraic formula for Cyc of polynomial-size by arithmetizing the Boolean circuit.

Furthermore, we have vertex variables $Y_1, \dots, Y_n$. Y_i will be set to 1 if the corresponding node is in the k-cycle and to 0 otherwise. This can be achieved by arithmetizing $\bigvee_{j=1}^{n} E_{j,i} \implies Y_i$. We can assume that $V_1 = \{1, \dots, n_1\}$ and $V_2 = \{n_1 + 1, \dots, n\}$. The k-cycle contributes weight $\prod_{i,j} E_{i,j} \cdot w(i, j)$. The uncovered nodes in V_1 contribute weight $\prod_{i=1}^{n_1} (1 - Y_i) w(i, i)$. The weight of the uncovered nodes in V_2 can be in principle computed using $\mathrm{per}^{(\leq c)}(G[V_2])$, which has a small circuit by Theorem 51 and even a polynomial-size formula by Remark 52. however, some nodes in V_2 may be covered by the k-cycle. Therefore, we replace every weight $w(i, j)$ by $(1 - Y_i) \cdot w(i, j)$ for $i \neq j$, turning each node i off that is in the k-cycle. Furthermore, we replace $w(i, i)$ by $(1 - Y_i) \cdot w(i, i) + Y_i$. This equips every node i that is turned off with a self-loop with weight 1, ensuring that it does not contribute to $\mathrm{per}^{(\leq c)}$. Altogether, we can write

$$\sum_{e,y \in \left\langle {n^2 \atop k} \right\rangle} \mathrm{Cyc}(e) \cdot \prod_{i,j} e_{i,j} \cdot w(i,j) \cdot \left[\bigvee_{j=1}^{n} e_{j,i} \implies y_i \right] \cdot \prod_{i=1}^{n_1} (1 - y_i) w(i,i) \cdot \mathrm{per}^{(\leq c)}(G'[V_2])$$

where $[\dots]$ denotes the arithmetization and G' is the graph with the modified weight functions as described above. ◀