Analysis of Algebraic Complexity Classes and Boolean Functions

By

Nitin Saurabh

MATH10201005003

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

(Theoretical Computer Science)

In partial fulfillment of requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

December, 2016

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Nitin Saurabh entitled "Analysis of Algebraic Complexity Classes and Boolean Functions" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

____ Date:

Chair - V. Arvind

____ Date:

Guide/Convener - Meena Mahajan

____ Date:

Examiner - Arkadev Chattopadhyay

____ Date:

Member 1 - Saket Saurabh

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date:

Place: Guide

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Nitin Saurabh

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Nitin Saurabh

List of Publications arising from the thesis

Journal

1. Some Complete and Intermediate Polynomials in Algebraic Complexity Theory. Meena Mahajan and Nitin Saurabh. To appear in Theory of Computing Systems – the special issue of CSR 2016.

2. VNP=VP in the multilinear world. Meena Mahajan, Nitin Saurabh, and Sébastien Tavenas. Information Processing Letters, **2016**, 116(2), pp 179-182.

3. Upper Bounds on Fourier Entropy. Sourav Chakraborty, Raghav Kulkarni, Satyanarayana V. Lokam, and Nitin Saurabh. Theoretical Computer Science, **2016**, vol. 654, pp 92-112 – the special issue of COCOON 2015.

4. Homomorphism Polynomials Complete for VP. Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Chicago Journal of Theoretical Computer Science, **2016**, 2016(3).

Conferences

1. Some Complete and Intermediate Polynomials in Algebraic Complexity Theory. Meena Mahajan and Nitin Saurabh. In Proceedings of the 11th International Computer Science Symposium in Russia (CSR), June 2016, St. Petersburg, Russia.

2. Upper Bounds on Fourier Entropy. Sourav Chakraborty, Raghav Kulkarni, Satyanarayana V. Lokam, and Nitin Saurabh. In Proceedings of the 21st International Computing and Combinatorics Conference (COCOON), August 2015, Beijing, China.

3. Homomorphism Polynomials Complete for VP. Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. In Proceedings of the 34th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), December 2014, New Delhi, India.

Nitin Saurabh

To my parents.

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Meena Mahajan wholeheartedly. She has been a constant source of inspiration. I will forever be grateful to her for her invaluable guidance and understanding since my undergraduate days. Over the years I have benefited immensely from discussions with her – technical or otherwise. Her thoughts and ideas have helped me shape my thought process as a researcher as well as grow as an individual. In short, I have been very fortunate to have her as my advisor.

I would also like to thank all the faculty members at IMSc, especially V. Arvind, Kamal Lodaya, Venkatesh Raman, R. Ramanujam, Saket Saurabh, Vikram Sharma and C. R. Subramanian, all of whom I have learnt a great deal from. I would especially like to thank them for their collective effort to provide an encouraging and fertile environment to learn and grow as a researcher and an individual. It has been an immensely satisfying learning process, and I am eternally grateful to have had them as mentors.

I also wish to thank all my collaborators. A special thanks to Satya Lokam, Valentine Kabanets, and Sourav Chakraborty. Discussions with them have taught me a great deal about many aspects of research. As a student I have had the good fortune to visit other academic institutes. In particular, I would like to thank Satya for sponsoring many visits to MSR Bangalore, Valentine for a visit to SFU Burnaby, and Kristoffer Hansen for a visit to Aarhus University.

I further take this opportunity to thank the faculty members at CMI. None of this would have started if not for the nurture that they provided during the formative years of my undergraduation.

I would like to thank my parents and my family members for their unwavering support that has allowed me to follow my dreams with freedom.

I also owe many thanks to all my friends. I have been fortunate to have a very diverse peer group who made sure that there never was a moment that lacked excitement.

Finally, I wish to thank IMSc administration for being helpful and providing excellent facilities to make life at IMSc very comfortable.

Contents

Synopsis

The thesis is divided into two parts, viz. Algebraic complexity theory and Boolean function analysis.

The first part deals with algebraic complexity theory, specifically with algebraic classes. First we study different kinds of reductions and prove lower bounds against them. In particular, we show that $\mathsf{sym\text{-}Perm}$ is VNP-complete over fields of characteristic other than 2. Then we prove that $\mathsf{Clique}^{\sqrt{n}}$ is not a monotone p-projection of Perm. We also show that multilinear algebraic classes are closed under exponential sums.

Next, we define and study polynomial families based on *graph homomorphisms*. Using these families we characterise the algebraic classes VBP, VP and VNP. We establish the first instance of natural families of polynomials that are defined independent of the circuit model, and are VP-complete. We further show the utility of homomorphism polynomials by exhibiting explicit polynomial families that are complete for VBP and VNP. Finally, we end the first part with a study of families of polynomials that are of *intermediate complexity*, that is, in VNP, but neither VNP-hard nor in VP unless PH collapses to the second level. Specifically, we exhibit a list of new natural VNP-intermediate families of polynomials that are defined using basic NP-complete problems.

In the second part of this thesis, we study the *Fourier Entropy-Influence (FEI) Conjecture*, made by Friedgut and Kalai in 1996. We start with establishing upper bounds on Fourier entropy of a Boolean function. These upper bounds are the combinatorial measures associated with a Boolean function that are known to be larger than the influence.

These complexity measures include, among others, the logarithm of the number of leaves and the average depth of a parity decision tree. We then show that for the class of Linear Threshold Functions (LTF), the Fourier Entropy is $O(\sqrt{n})$. It is known that the average sensitivity for the class of LTF is $\Theta(\sqrt{n})$. We also establish a bound of $O_d(n^{1-\frac{1}{4d+6}})$ for general degree-d polynomial threshold functions. Next we proceed to show that the FEI Conjecture holds for read-once formulas that use AND, OR, XOR, and NOT gates. Finally, we give a general bound involving the first and second moments of sensitivities of a function (average sensitivity being the first moment), which holds for real-valued functions as well.

List of Figures

Chapter 1

Introduction

Complexity theory aims to understand the power and limitations of “efficient” computation. In other words, a simple goal is to quantify the computational resources – time, space, queries, randomness, etc. – required to solve a given task. The progress in our understanding of computation, in particular “efficient” computation, led to the discovery of numerous natural problems that were inherently computational in nature. Some of them turned out to have an “easy” solution, and some resisted all attempts to be solved “easily”. This resulted in a formalized notion of computational efficiency, giving rise to the famous P vs NP problem.

Despite decades of effort, we found ourselves in a difficult situation where we had many questions of fundamental importance, but very few answers (and still remains unanswered). However, we made significant progress in our understanding of restricted model of computation, in particular restricted classes of Boolean circuits. Inspired from the progress in restricted setting, Valiant [Val92] wondered about pursuing different approaches that might contribute to progress on the unrestricted model. In particular, he argued, “If $\mathsf{P} \neq \mathsf{NP}$ then any circuit-theoretic proof of this would have to be preceded by analogous results for the more constrained arithmetic model.”

The purpose of this thesis is two fold. First, to make progress in our understanding of "efficient" computation in the algebraic model of computation, and second, to further our understanding of Boolean functions using tools from Fourier analysis.

In particular, the thesis is divided into two parts. The *first* part deals with Algebraic Complexity theory. Here we study different kinds of reductions and prove lower bounds against them. We further define and study *homomorphism polynomials* and use them to characterise the algebraic classes VBP, VP, and VNP. In particular, we define *natural* polynomials that are VP-complete under the strictest notion of reduction. The importance of these polynomials stems from the fact that they are the first such polynomials, that are defined independent of the circuit model, and shown to be VP-complete. We end the first part with a study of polynomial families that are of *intermediate* complexity, i.e., in VNP, but, under certain complexity theoretic assumption, neither VNP-hard, nor in VP.

The *second* part of the thesis is an attempt to solve the *Fourier Entropy-Influence Conjecture*, made by Friedgut and Kalai [FK96] in 1996. Resolving the conjecture is one of the most important open problems in the Fourier analysis of Boolean functions [Kal]. Here we establish the conjecture for *Read-Once* formulas over AND, OR, XOR, and NOT gates. Furthermore we provide (various) upper bounds on *Fourier entropy* of general Boolean functions and polynomial threshold functions, that may be viewed as progress towards the Fourier Entropy-Influence conjecture.

Below we state these problems in detail with some background and motivation, and mention our contribution.

1.1 Algebraic Complexity Theory

Algebraic Complexity theory is the study of computation of families of polynomials using the underlying field (or, ring) operations. In this thesis, we will only be concerned with polynomial families (f_n) such that both, the number of variables and the degree of f_n, are bounded by polynomial functions in n. Further we use the notion of *projections* to compare two families of polynomials.

A polynomial f is a *projection* of a polynomial g if for some $\sigma_i \in \mathbb{F} \cup \{x_1, \ldots, x_n\}$, $f(x_1, \ldots, x_n) = g(\sigma_1, \ldots, \sigma_m)$. Further, a sequence of polynomials (f_n) is a *p-projection* of the family (g_n) if f_n is a projection of $g_{t(n)}$ for some polynomial function $t(\cdot)$. (If $t(n)$ grows like $2^{\mathsf{poly}(\log n)}$, then we call it a qp-projection.) So essentially projection is a way to represent one polynomial using another. A motivation to study such a restrictive kind of reduction is that it easily transfers computational hardness among polynomial families.

Reductions and Lower bounds

The representation of polynomials using the determinant is a classical subject [Gra55, Sch81, Dic21, CT79]. A natural restriction in the study of the representation of polynomials is to condition the matrix to be symmetric. That is, a symmetric matrix A with entries in $\mathbb{F} \cup \{x_1, \ldots, x_n\}$ such that $f = \mathsf{det}(A)$. Due to the widespread applications of the determinant, there has been a long line of work, from as early as the nineteenth century, on *symmetric determinantal representations* of polynomials [Hes55, Cay69, Dix02, Dic21, Cat81, Bea00, HMV06, HV07, Brä11, GKKP11, PSV11, NT12, Qua12, NPT13, Brä13, GMT13]. In recent years, the study of representations using symmetric determinant has received impetus due to its importance in convex optimization.

We study the representation of polynomials by the permanent of a symmetric matrix. That is, we will represent polynomials as the permanent of an undirected simple graph.

Formally, let $X_n = [x_{i,j}]_{1 \leqslant i,j \leqslant n}$ be an $n \times n$ symmetric symbolic matrix, that is $x_{i,j} = x_{j,i}$. Then, $\mathsf{sym\text{-}Perm} = (\mathsf{sym\text{-}Perm}_n)$ is a family of polynomials where $\mathsf{sym\text{-}Perm}_n$ is the permanent of the matrix X_n.

We study the following question: *Over fields of characteristic not equal to 2, is every family in* VNP *a p-projection of* sym-Perm *?*

In other words, is sym-Perm VNP-hard over fields of characteristic different than 2, with respect to p-projections?

From the works on factorization of polynomials [Kal86, Kal87, Kal89, KT90], it follows (see [Bür00a]), that over fields of char 0, $\mathsf{sym\text{-}Perm}$ is VNP-hard with respect to c-reductions. c-reductions are the algebraic analogue of oracle reductions (see Definition 2.3.4). Moreover, using the recent results of Oliveira [Oli15], on factoring polynomials with low individual degree, the reduction above can be improved to *constant-depth* c-reductions (see Definition 2.3.5).

A further restriction of constant-depth c-reductions is the *linear p-projection*, where each f_n is a linear combination $\sum_k \lambda_k g_{i_k}$ of polynomially many p-projections of g (see Definition 2.3.1).

In Chapter 2, we show that Perm_n can be written as a difference of two projections of $\mathsf{sym\text{-}Perm}_{10n}$. Hence we get the following result.

Result 1. *Over fields of characteristic not equal to 2,* $\mathsf{sym\text{-}Perm}$ *is* VNP*-complete with respect to linear p-projections. Furthermore, there are only two summands in the linear p-projection.*

It remains open whether $\mathsf{sym\text{-}Perm}$ is VNP-hard with respect to p-projections. Observe that bringing down the number of summands from 2 to 1, in Result 1, will establish hardness under p-projections.

In Chapter 2, we also prove lower bounds against *monotone* projections. When the underlying field is an ordered field, such as $\mathbb{Q}$ and $\mathbb{R}$, or, more generally, any totally ordered semi-ring, such as Boolean $\{\wedge, \vee\}$-semi-ring, a projection is called *monotone* if and only if all constants appearing in the substitution are *non-negative*.

Razborov [Raz85a] proved that computing the permanent, over the Boolean $\{\wedge, \vee\}$-semi-ring, requires monotone circuits of size at least $n^{\Omega(\log n)}$. Till date, this is the best lower bound known over the Boolean $\{\wedge, \vee\}$-semi-ring. Jukna [Juk14] observed that if the family of the Hamiltonian cycle polynomial is a *monotone* p-projection of the permanent family, over the Boolean $\{\wedge, \vee\}$-semi-ring, then, via the Alon-Boppana lower bound for clique [AB87], one would get a lower bound of $2^{n^{\Omega(1)}}$ for monotone circuits computing Perm_n, thus improving on [Raz85a]. It is also worthwhile to note that such a monotone p-projection, over $\mathbb{R}$, would give an alternate proof of the fact that computing permanent by monotone circuits over reals requires size at least $2^{n^{\Omega(1)}}$. (A stronger version of this fact was proved by [JS82].)

Grochow, in [Gro15], made progress on Jukna's observation by establishing a formal connection between *monotone projections* and *extended formulations* of linear programs. Using this he showed that the Hamiltonian cycle polynomial is not a monotone *sub-exponential-size* projection of the permanent. Though it answered Jukna's specific question about the Hamiltonian cycle in its entirety, the underlying motivating question still remains unanswered : *Whether clique is a monotone p-projection of the permanent?* May be not via the Hamiltonian cycle polynomial, but perhaps via something else, say, via the *satisfiability* polynomial [Val79]. It is known (see Section 5 [AB87]) that clique is a monotone projection of the satisfiability polynomial. Thus it still left open the possibility of transferring monotone circuit lower bounds for clique to the permanent.

Here we answer the main motivating question of Jukna by directly proving that the $\mathsf{Clique}^{\sqrt{n}} = (\mathsf{Clique}_n^{\sqrt{n}})$ family is not a monotone (affine) polynomial-size projection of Perm. By $\mathsf{Clique}_n^{\sqrt{n}}$ we mean the polynomial which enumerates $\sqrt{n}$-sized cliques in an n-vertex graph.

Result 2. *Over the reals (or any totally ordered semi-ring), the* $\mathsf{Clique}^{\sqrt{n}}$ *family is not a monotone (affine) p-projection of the* Perm *family. In fact, if* $\mathsf{Clique}_n^{\sqrt{n}}$ *is a monotone (affine) projection of* $\mathsf{Perm}_{t(n)}$*, then* $t(n) \geqslant 2^{\Omega(\sqrt{n})}$.

Thus this possibility of transferring monotone circuit lower bounds for clique to permanent cannot work. Our proof strategy is similar to [Gro15], that is, it uses the connection between monotone projections and extended formulations. We further establish that certain non-negative polynomials (i.e., polynomials with non-negative coefficients), such as $\mathsf{Sat}^{\mathsf{q}}$ and $\mathsf{Clow}^{\mathsf{q}}$, are not monotone p-projections of Perm. We will describe these polynomials later, in more detail, when we study polynomial families with *intermediate* complexity.

Closure under exponential sums

A sequence of polynomials (f_n) belongs to VNP if and only if there exist polynomials p and q, and a sequence $(g_n) \in \mathsf{VP}$ such that for all n,

$$f_n(x_1, \ldots, x_{q(n)}) = \sum_{\bar{y} \in \{0,1\}^{p(n)}} g_n(x_1, \ldots, x_{q(n)}, y_1, \ldots, y_{p(n)}).$$

So, in other words, one can think of VNP as *exponential sums* of polynomial sized circuits; $\mathsf{VNP} = \sum \cdot \mathsf{VP}$. Hence the VP versus VNP question can also be thought of as understanding the power of exponential sums. In the foundational paper [Val82], Valiant observed that exponential sums of polynomial sized formulas ($\sum \cdot \mathsf{VF}$) exactly capture exponential sums of polynomial sized circuits ($\sum \cdot \mathsf{VP}$). He used this observation crucially to show that the permanent family Perm is VNP-hard. Hence from Valiant's observation, it follows that $\sum \cdot \mathsf{VF} = \sum \cdot \mathsf{VP} = \mathsf{VNP}$.

Valiant's observation raises a natural question to study: How powerful are *exponential sums* of *restricted* circuit classes?

A natural restriction on arithmetic circuits is *multilinearity*. A polynomial is called *multilinear* if each variable in the polynomial has degree at most 1. An arithmetic circuit is called *multilinear* if every gate in it computes a multilinear polynomial. Furthermore, if for every product gate, the sub-circuits rooted at the left and right child are variable-disjoint, then the circuit is called *syntactic multilinear*.

The exponential summation under the restriction of syntactic multilinearity was studied by Jansen et al. [MR08, JR09, JMR13]. They showed that syntactic multilinear classes are closed under exponential sums. Contrast this with the case of general formulas, where it captures VNP. Exponential summations of polynomials were also studied by Juma et al. [JKRS09]. Their motivation was to obtain query algorithms for #SAT that are better than brute-force. They proved that over fields of characteristic different from 2, multilinear polynomials are closed under exponential sums.

We end Chapter 2 with a study of the exponential summation under the restriction of *multilinearity* (not necessarily syntactic). Using techniques different from those used in [JMR13, JKRS09], we extend their results by showing that, over any field, exponential summation does not add power to multilinear circuit classes.

Result 3. *Let $f(x_1,\ldots,x_N,y_1,\ldots,y_m)$ be a polynomial that is multilinear in the $Y = \{y_1,\ldots,y_m\}$ variables. Let $h(X)$ be the exponential sum polynomial*

$$h(X) = \sum_{e\in\{0,1\}^m} f(X,e_1,\ldots,e_m).$$

If f has an efficient computation, so does h. The following table gives upper bounds on the complexity measures of h in terms of the corresponding measures of f.

		Char $\neq 2$	*Char* $= 2$	
			infinite fields	*finite fields*
Circuit (size,width)	f	s, w	s, w	s, w
	h	$s+1, w$	$3s(m+1), w+1$	$s(m+1)^2, w(m+1)$
ABP (size,width)	f	s, w	s, w	s, w
	h	$s+1, w$	$3s(m+1), w+2$	$s(m+1), w(m+1)$
Formula size	f	s	s	s
	h	$s+1$	*$O(s)$ [JMR13]*	*$O(s)$ [JMR13]*

Furthermore, if the circuit/ABP[1]/formula for f is multilinear, then so is the circuit/ABP/formula for h.

[1] ABP stands for Algebraic Branching Program. For a definition, see Definition 2.2.2.

Essentially, the above result says, an exponential summation over multilinear variables is as good as evaluating the polynomial at one or a small number of points. As a corollary we obtain the closure property for numerous multilinear classes (Corollary 2.6.4). In particular, a corollary of our result is that $\mathsf{VNP} = \mathsf{VP}$ in the multilinear setting, whereas we do not believe that a similar thing holds in non-multilinear setting. Indeed, our result, along with the fact VF is strictly weaker than VBP[2] ([Raz06, DMPY12]), implies that in the multilinear world we do not have an analogue of the collapse $\sum \cdot \mathsf{VF} = \sum \cdot \mathsf{VBP} = \sum \cdot \mathsf{VP}$ that holds in the general world. Thus our result highlights essential differences between the general and multilinear worlds, and indicates that separations/collapses in the restricted multilinear world may have no bearing on the true state of affairs in the general world.

The results described here are either unpublished or appear in [MS16, MST16].

Completeness

Valiant [Val79, Val82] developed the theory of completeness in the algebraic model of computation. He showed the permanent family Perm to be complete for the class VNP (over char $\neq 2$), and the determinant family Det to be complete for VP under qp-projections. Hence, the VP vs VNP problem became synonymous with Perm vs Det problem. In other words, can the permanent of a matrix be written down as the determinant of a matrix of not too large a dimension? This reformulation became significant for two reasons: one, the Perm vs Det question is a purely algebraic statement devoid of any model of computation, and two, combinatorialists have long been fascinated by this problem [Pol13, Sze13, MM60, MM61, Min78].

However, the reformulation left a puzzling scenario. While we know that Perm is VNP-complete under p-projections, it is *not* known whether Det is VP-complete under p-

[2] VBP is the class of polynomial families computable by polynomial size ABPs. Also, see Definition 2.2.5.

projections. In fact, with respect to p-projections, the determinant family is complete for the possibly smaller class VBP of polynomial-sized algebraic branching programs (ABPs).

This raises an important and interesting question: *Are there 'natural'* VP*-complete polynomial families?*

The very first polynomial shown to be VP-complete, in [vzG87], was motivated by the definition of VP. Indeed the polynomials were so constructed that every polynomial of degree at most n over n variables is a projection of the n-th polynomial in the family. von zur Gathen [vzG87] explicitly stated the question of finding "natural" families that are VP-complete. Then, in [Bür00a], Bürgisser showed that a generic polynomial family constructed recursively while controlling the degree is complete for VP. The construction directly follows a topological sort of a generic VP circuit. In [Raz10] (see also [SY10]), Raz used the depth-reduction of [VSBR83] to show that a family of "universal circuits" is VP-complete; any VP computation can be embedded into it by appropriately setting the variables. All three of these VP-complete families are thus directly obtained using the circuit definition / characterization of VP.

In Chapter 3, we define and study *homomorphism polynomials*. Using homomorphism polynomials, we establish the first instance of natural families of polynomials that (1) are defined independently of the circuit definition of VP, and (2) are VP-complete.

We first set up some notation. Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two graphs. Let $\alpha : V(G) \to \mathbb{N}$ be a labeling of vertices of G by non-negative integers. Consider the set of variables $\overline{X} := \{X_u \mid u \in V(H)\}$ and $\overline{Y} := \{Y_{(u,v)} \mid (u,v) \in E(H)\}$. The *weighted* homomorphism polynomial $f^{\alpha}_{G,H}$ in the variable set $\overline{X} \cup \overline{Y}$ is defined as follows:

$$f^{\alpha}_{G,H} = \sum_{\phi \in \mathbf{Hom}} \left(\prod_{u \in V(G)} X^{\alpha(u)}_{\phi(u)} \right) \left(\prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))} \right),$$

where **Hom** is the set of all homomorphisms from G to H (adjacencies preserving maps from $V(G)$ to $V(H)$). Moreover, for our purposes, $\{0,1\}$-valued weights suffices, i.e., $\alpha : V(G) \to \{0,1\}$. Such $\{0,1\}$-valued weights are commonly used in the literature, see, e.g., [BCL+06]. To obtain families of polynomials from the homomorphism polynomial we consider sequences of graphs (G_m) and (H_m).

Result 4. *Over fields of characteristic 0, the family of homomorphism polynomials (f_m), with $f_m(\overline{X}, \overline{Y}) = f^{\alpha}_{G_m,H_m}(\overline{X}, \overline{Y})$, where*

- *$G_m := \mathsf{T}_m$, where T_m denotes the complete (perfect) binary tree with m leaves.*
- *H_m is an undirected complete graph on $\mathsf{poly}(m)$, say m^6, nodes.*
- *Define $\alpha : \mathsf{T}_m \to \{0,1\}$ such that,*

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

is complete for VP *with respect to linear p-projections.*

We further improve on our Result 4 by establishing VP-completeness (1) for a much simpler polynomial, and (2) with respect to p-projections. Consider the following *homomorphism polynomial* defined only over the variable set $\overline{Y}$:

$$f_{G,H} = \sum_{\phi \in \mathbf{Hom}} \left(\prod_{(u,v)\in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

We construct a sequence (G_m) of *bounded* tree-width graphs such that $(f_{G_m,H_m}(\overline{Y}))$ is complete for VP under p-projections. We use *rigid* and mutually *incomparable* graphs in the construction of G_m. A graph is called *rigid* if it has *no* homomorphism to itself other than the identity map. Two graphs G and H are called *incomparable* if there are *no* homomorphisms from $G \to H$ as well as $H \to G$.

Result 5. *Over any field, the family of homomorphism polynomials* (f_m)*, with* $f_m(\overline{Y}) = f_{G_m,H_m}(\overline{Y})$*, where*

- G_m *is obtained from* T_m *by "replacing" nodes with rigid and mutually incomparable graphs and "stretching" edges (of the tree) into long paths, and*
- H_m *is an undirected complete graph on* $\mathsf{poly}(m)$*, say* m^6*, vertices,*

is complete for VP *under p-projections.*

Moreover, based on homomorphism polynomials, we obtain a characterisation of VP, VBP, and VNP. A sequence (G_m) of graphs is called a p-family if the number of vertices in G_m is bounded by a polynomial function of m.

Result 6. *Let* (G_m) *and* (H_m) *be p-families of graphs. Consider the family of homomorphism polynomials* $f = (f_m)$*, where* $f_m(\overline{Y}) = f_{G_m,H_m}(\overline{Y})$*. Then,*

- $f \in \mathsf{VNP}$*. Furthermore, there exists an explicit p-family* (G_n) *of graphs where* G_n *has tree-width* $\Theta(n)$*, and* H_n *is a complete graph on* $O(n^4)$ *vertices, such that* (f_{G_n,H_n}) *is* VNP*-hard, over any field, with respect to p-projections.*
- *If the sequence* (G_m) *has bounded tree-width,* $f \in \mathsf{VP}$*. Furthermore, there exists an explicit p-family* (G_n) *of bounded tree-width graphs, and* H_n *is a complete graph on* $O(n^6)$ *vertices, such that* (f_{G_n,H_n}) *is* VP*-hard, over any field, with respect to p-projections.*
- *If the sequence* (G_m) *has bounded path-width,* $f \in \mathsf{VBP}$*. Furthermore, there exists an explicit p-family* (G_n) *of bounded path-width graphs, and* H_n *is a complete graph on* $O(n^2)$ *vertices, such that* (f_{G_n,H_n}) *is* VBP*-hard, over any field, with respect to p-projections.*

Our upper bounds, in particular of VP and VBP, are obtained in an uniform way. The construction is inspired from dynamic programming on *nice* tree decomposition of graphs.

It also gives an alternate proof of the fact that every polynomial family in VP has *skew* circuits of size $2^{O(\log^2 n)}$ (see also [MP08]).

The hardness results are established by showing that parse trees (or, s-t-paths) in the universal circuit in a normal form (or, a generic ABP) can be captured by homomorphisms from a "tree"-like (or, a "path"-like) structure G_n. We use projections to obtain the generic circuit (or, ABP) from the graph H_n. The rigid and mutually incomparable graphs ensure that there is a bijection between surviving homomorphisms and parse trees. Since parse trees (or, s-t-paths) account for all monomials generated by the circuit (or, ABP), we obtain a generic VP (or, VBP) polynomial as a projection of our homomorphism polynomials.

In the context of Result 6, a very natural question falls out: *What is the complexity of a family of homormophism polynomials, where the sequence* (G_n) *is such that* G_n *has tree-width* $\mathrm{o}(n)$*, say* $\mathsf{poly}(\log n)$*?*

Consider the family Clique_n^k, where

$$\mathsf{Clique}_n^k := \sum_{\substack{S\subseteq[n]\\|S|=k}} \prod_{\substack{i,j\in S\\i<j}} x_{i,j}.$$

It enumerates k-sized cliques in an n-vertex graph. Set $k = \log n$. From our upper bound, it follows that $\mathsf{Clique}_n^{\log n}$ is computable by an arithmetic circuit of size $n^{O(\log n)}$. Consequently, if $\mathsf{Clique}^{\log n}$ is VNP-complete then all families in VNP will have $n^{O(\log n)}$-sized circuits computing them. This contradicts *Valiant's extended hypothesis* $\mathsf{VNP} \neq \mathsf{VQP}$. Such observations motivated us to look at polynomials that have *intermediate* complexity.

The results described here appear in [DMM+16, MS16].

Intermediate complexity

Let us call a polynomial family VNP-intermediate if it is (1) in VNP, (2) not VNP-complete, and (3) not in VP.

Inspired from classical results in structural complexity theory, in particular [Lad75], Bürgisser [Bür99] proved that if Valiant's hypothesis (i.e. VP $\neq$ VNP) is true, then, over any field there is a p-family in VNP which is neither in VP nor VNP-complete with respect to c-reductions. Further, Bürgisser [Bür99] showed that over finite fields, a *specific* family of polynomials is VNP-intermediate, provided the polynomial hierarchy PH does not collapse to the second level. Informally, these polynomials enumerate *cuts* in a graph. This is a remarkable result, when compared with the classical P-NP setting or the BSS-model. Though the existence of problems with intermediate complexity has been established in the latter settings, due to the involved "diagonalization" arguments used to construct them, these problems seem highly unnatural. That is, their definitions are not motivated by an underlying combinatorial problem but guided by the necessities of the proof. The question of whether there are other naturally-defined VNP-intermediate polynomials was left open by Bürgisser [Bür00a]. We remark that until this work the *cut enumerator* polynomial from [Bür99] was the only known example of a natural polynomial family that is VNP-intermediate.

In Chapter 4, we provide a list of new natural VNP-intermediate polynomial families, based on basic (combinatorial) NP-complete problems.

For a fixed finite field $\mathbb{F}_q$ of size q and char p, consider the following families.

(1) The *satisfiability* polynomial $\mathsf{Sat}^{\mathsf{q}} = (\mathsf{Sat}^{\mathsf{q}}{}_n)$: For each n, let Cl_n denote the set of all possible clauses of size 3 over $2n$ literals. There are n variables $\tilde{X} = \{X_i\}_{i=1}^n$, and also $8n^3$ clause-variables $\tilde{Y} = \{Y_c\}_{c\in\mathsf{Cl}_n}$, one for each 3-clause c.

$$\mathsf{Sat}^{\mathsf{q}}{}_n := \sum_{a\in\{0,1\}^n}\left(\prod_{i\in[n]:a_i=1} X_i^{q-1}\right)\prod_{\substack{c\ \in\mathsf{Cl}_n\\ a \text{ satisfies } c}} Y_c^{q-1}\ .$$

(2) The *clow* polynomial $\mathsf{Clow}^{\mathsf{q}} = (\mathsf{Clow}^{\mathsf{q}}{}_n)$: For each n, let G_n denote the complete graph on n nodes. A clow in an n-vertex graph is a closed walk of length exactly n, where the minimum numbered vertex (called the head) appears exactly once. The set of variables are $\tilde{X} = \{X_e\}_{e \in E_n}$ and $\tilde{Y} = \{Y_v\}_{v \in V_n}$.

$$\mathsf{Clow}^{\mathsf{q}}{}_n := \sum_{w:\text{ clow of length } n} \left(\prod_{e:\text{ edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v:\text{ vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

Similarly, we define polynomials based on other combinatorial problems, e.g., *vertex cover* polynomial VC^{q}, *clique/independent set* polynomial $\mathsf{CIS}^{\mathsf{q}}$, and *3D-matching* polynomial $\mathsf{3DM}^{\mathsf{q}}$. We show that under the plausible hypothesis $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, all five polynomials mentioned above are VNP-intermediate.

Result 7. *Over a finite field $\mathbb{F}_q$ of characteristic p, the polynomial families* $\mathsf{Sat}^{\mathsf{q}}$, VC^{q}, $\mathsf{CIS}^{\mathsf{q}}$, $\mathsf{Clow}^{\mathsf{q}}$, *and* $\mathsf{3DM}^{\mathsf{q}}$, *are in* VNP. *Further, if* $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, *then they are all* VNP-*intermediate; that is, neither in* VP *nor* VNP-*hard with respect to c-reductions.*

The results described here appear in [MS16].

1.2 Analysis of Boolean functions

Boolean functions are one of the most fundamental object of study in computer science. Within theoretical computer science, Fourier analysis of Boolean functions evolved into one of the most useful and versatile tools to study such functions (see [dW08, O'D14]).

The set of all real functions on $\{0,1\}^n$ is a 2^n-dimensional real vector space with an inner product defined by $\langle f, g \rangle = 2^{-n} \sum_{x \in \{0,1\}^n} f(x)g(x)$. The character functions $\chi_S(x) := (-1)^{\sum_{i \in S} x_i}$ for $S \subseteq [n]$ form an orthonormal basis for this space of functions with respect to the above inner product. Thus, every function $f : \{0,1\}^n \to \mathbb{R}$ has the *unique Fourier expansion*: $f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S(x)$.

For a Boolean function $f\colon \{0,1\}^n \to \{+1,-1\}$, we have $\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1$. Hence we can define the (Shannon) entropy of the distribution given by $\widehat{f}(S)^2$:

$$\mathbb{H}(f) := \sum_{S\subseteq[n]} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2}.$$

The *influence of f in the i-th direction* $\mathsf{Inf}_i(f)$ is the fraction of inputs at which the value of f gets flipped if we flip only the i-th bit. Then the (total) *influence* $\mathsf{Inf}(f)$ of f is $\sum_{i=1}^n \mathsf{Inf}_i(f)$.

The Fourier Entropy-Influence (FEI) Conjecture, made by Friedgut and Kalai [FK96], states that for every Boolean function, its Fourier entropy is bounded above by its total influence.

Fourier Entropy-Influence Conjecture: There exists a universal constant C such that for all $f : \{0,1\}^n \to \{+1,-1\}$, $\mathbb{H}(f) \leqslant C \cdot \mathsf{Inf}(f)$.

The original motivation for the conjecture stems from a study of *threshold phenomena* in random graphs. Friedgut and Kalai [FK96] asked : *How large can the threshold interval be for a monotone graph property?*

Consider $f : \{0,1\}^n \to \{0,1\}$ representing a monotone graph property. Define $A_f(p) := \Pr[f(X_1, X_2, \ldots, X_n) = 1]$, where X_i's are independent random variables, and each X_i is 1 with probability p and 0 with probability $1-p$. Let $\delta > 0$ be a small number. By threshold interval we mean the length of the interval $[p,q]$ such that $A_f(p) = \delta$, but $A_f(q) = 1-\delta$. Then, the length of the threshold interval is inversely proportional to the derivative of $A_f(p)$, and by Russo's formula [Rus81, Mar74], the derivative of $A_f(p)$ equals the total influence of f (under the product measure). Hence, the graph property has a small threshold interval around p, that is, sharp threshold, if and only if it has large influence (under the product measure). Therefore, Friedgut and Kalai [FK96] asked for generic conditions that would force the influence to be large. They conjectured that a spread-out Fourier spectrum, i.e. large Fourier entropy, might be one such condition.

The first progress on the FEI conjecture was made by Klivans et al. [KLW10] showing that the conjecture holds for random DNFs. O'Donnell et al. [OWZ11] proved that the conjecture holds for symmetric functions and, more generally, for any d-part symmetric functions for constant d. They also established the conjecture for functions computable by read-once decision trees. Keller et al. [KMS12] studied a generalization of the conjecture to biased product measures on the Boolean cube and proved a variant of the conjecture for functions with extremely low Fourier weight on high levels. O'Donnell and Tan [OT13] verified the conjecture for read-once formulas using a composition theorem for the FEI conjecture. Wan et al. [WWW14] studied the conjecture from the point of view of existence of efficient prefix-free codes for the random variable, $\mathcal{X} \sim \widehat{f}^2$, that is distributed according to $\widehat{f}^2$. Using this interpretation, they verified the conjecture for bounded-read decision trees.

In Chapter 5, we study the Fourier Entropy-Influence (FEI) conjecture, and report various upper bounds on the Fourier entropy of Boolean functions and general real-valued functions. Further, we prove the conjecture for *Read-Once* formulas.

The $\mathsf{Inf}(f)$ of a Boolean function f lower bounds a number of complexity parameters of f such as average depth of a decision tree that computes f (see Fig. 5.1). Hence a natural weakening of the FEI conjecture is to prove upper bounds on the Fourier entropy in terms of such complexity measures of Boolean functions.

It is easy to observe $\mathbb{H}(f) = O(\log L_1(f))$, and thus several easier bounds from Fig. 5.1 follows. In particular, it implies $\mathbb{H}(f) = O(\log L(f))$, where $L(f)$ denotes the minimum number of leaves in a decision tree that computes f. If $\mathsf{DNF}(f)$ denotes the minimum size of a DNF for the function f, note that $\mathsf{DNF}(f) \leqslant L(f)$. It follows that improving the aforementioned upper bound on entropy to $O(\log \mathsf{DNF}(f))$ would resolve *Mansour's conjecture* – a long-standing open question about sparse Fourier approximations to DNF formulas motivated by applications to learning theory – and a special case of the FEI conjecture for DNF's. We prove the following upper bound of average depth.

Result 8. *For every Boolean function f, $\mathbb{H}(f) \leqslant 2 \cdot \oplus\text{-}\bar{\mathsf{d}}(f)$, where $\oplus\text{-}\bar{\mathsf{d}}(f)$ denotes the minimum average depth of a parity decision tree computing f. Moreover, the constant 2 in the bound is optimal.*

We next study the FEI conjecture for special classes of Boolean functions, namely threshold functions and Read-Once formulas.

It is known that the influence for the class of linear threshold functions is $\Theta(\sqrt{n})$, and for degree-d polynomial threshold functions is $O_d(n^{1-(1/4d+6)})$ [HKM14, DRST14]. This suggests a natural and important weakening of the FEI conjecture: *Is Fourier Entropy of polynomial threshold functions bounded by a similar function of n as their influence?* We study the *derivative* of *noise sensitivity*, and prove a technical lemma that bounds the derivative in terms of a noise parameter. Using this bound, we answer the above question positively.

Result 9. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a Boolean function. Then,*

- *If f is a linear threshold function, $\mathbb{H}(f) \leqslant C \cdot \sqrt{n}$, where C is a universal constant.*
- *If f is a degree-d polynomial threshold function, $\mathbb{H}(f) \leqslant C \cdot 2^{O(d)} \cdot n^{1-\frac{1}{4d+6}}$, where C is a universal constant.*

We further prove that the FEI conjecture holds for Read-Once formulas over AND, OR, NOT, and XOR gates. Our result is independent of a concurrent result by O'Donnell and Tan [OT13] that proves the FEI conjecture holds for read-once formulas that allow *arbitrary* gates of bounded fan-in. Prior to these results, O'Donnell et al. [OWZ11] proved that the FEI conjecture holds for read-once *decision trees*. Our result for read-once formulas is a strict generalization of their result. For instance, the tribes function is computable by read-once formulas but not by read-once decision trees.

Result 10. *If f is computed by a read-once formula using AND, OR, XOR, and NOT gates, then $\mathbb{H}(f) \leqslant 10\,\mathsf{Inf}(f)$.*

We end Chapter 5 with a study of real-valued functions. We prove an upper bound on the entropy of real-valued functions, and present some examples that throw light on how Boolean-ness is important for the veracity of the Fourier entropy-influence conjecture.

In particular, motivated by the following equivalent restatement of the FEI conjecture, we establish Result 11.

Fourier Entropy-Influence conjecture (equivalent): There is an absolute constant C such that for all Boolean function f, $\mathbb{H}(f) \leqslant C \cdot \sum_S |S| \widehat{f}(S)^2$.

Result 11. *If $f = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S$ is a real-valued function on the domain $\{0,1\}^n$ such that $\sum_S \widehat{f}(S)^2 = 1$, then, for any $\delta \in (0,1]$,*

$$\mathbb{H}(f) \triangleq \sum_{S \subseteq [n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant \sum_{S \subseteq [n]} |S|^{1+\delta} \widehat{f}(S)^2 + (\log n)^{O(1/\delta)}.$$

The results described here appear in [CKLS15].

1.3 Organisation of the thesis

The rest of the thesis is divided into *two* parts and *five* chapters. The first part of the thesis deals with the algebraic complexity theory (Chapter 2, 3, and 4). The second part deals with the analysis of Boolean function (Chapter 5).

In the *second* chapter, we study the sym-Perm polynomial family, lower bounds against *monotone projections*, and closure under *exponential sums*. The *third* chapter is devoted to the VP-completeness and *characterisation* of the algebraic complexity classes. Next, in the *fourth* chapter, we study families of *intermediate* complexity, and establish VNP-intermediate families.

We study the Fourier Entropy-Influence conjecture in the *fifth* chapter. Finally we conclude the thesis in the *sixth* chapter.

Part I

Algebraic Complexity Theory

Chapter 2

Structure of algebraic complexity classes

2.1 Introduction

The theory of NP-completeness is of fundamental significance in computational complexity. Valiant [Val79, Val82] developed analogous concept in the framework of algebraic complexity theory, and argued [Val92] that corresponding concepts in this setting must be understood, before we can give a satisfactory answer in Boolean setting. In [Val79], he proposed a theory of P vs NP in the algebraic setting. Analogously he defined and studied two classes of polynomial families, namely *p-computable* and *p-definable*. They are now known as VP and VNP, respectively. Further, to facilitate reductions among problems in the algebraic setting he studied *projections*. It is the strictest notion of reduction whereby one polynomial is obtained from another by simple substitutions. Roughly, families in VNP can be thought as sums of a family in VP over all possible Boolean substitutions of a (fixed) subset of variables [Val82]. We call such sums over all possible Boolean instantiations *exponential sums*. He also proved that the permanent family is VNP-complete under projections. Further, in [Val82], he studied closure properties of algebraic classes, and proved VNP to be closed under many natural operations such as substitution, coefficient extraction, differentiation, integration, etc.

In this chapter we try to understand different notions of reductions and aim to establish VNP-completeness of the family of the permanent of symmetric symbolic matrices. Valiant's [Val79] proof of the completeness of permanent crucially uses (non-symmetric) directed graphs. We further study the closure property of algebraic classes under *exponential sums*.

We give basic definitions about algebraic complexity classes and set up the general background in Section 2.2 and Section 2.3. We then study VNP-completeness of the symmetric permanent family under different kinds of reductions in Section 2.4. In Section 2.5, we prove lower bounds against *monotone* projections. Further, in Section 2.6, we study (exponential) sums of restricted algebraic classes under (partial) Boolean substitutions.

2.2 Preliminaries

We start with formal definitions in the setting of algebraic complexity. Let $\mathbb{F}$ be any field, and let $\mathbb{F}[x_1, \ldots, x_n]$ be the ring of polynomials over indeterminates $x_1, \ldots, x_n$ with coefficients from $\mathbb{F}$. We call a function $t : \mathbb{N} \to \mathbb{N}$ *p-bounded* if and only if there exists some $c > 0$ such that $t(n) \leqslant n^c + c$ for all n. It is called *qp-bounded* when $t(n) \leqslant 2^{c \cdot \log^c n}$ for all n.

The objects of our study will be families of polynomials $(f_n)_{n \geqslant 1}$ such that $f_n \in \mathbb{F}[x_1, \ldots, x_{v(n)}]$ for some function $v \colon \mathbb{N} \to \mathbb{N}$. Furthermore, if both the degree of f_n, and the number of variables $v \colon \mathbb{N} \to \mathbb{N}$ are p-bounded functions of n, we say (f_n) is a *p-family*. In this thesis, we will only concern ourselves with p-families of sequence of polynomials. The complexity measure of our interest would be the number of ring operations, additions and multiplications, needed to compute a polynomial symbolically.

Definition 2.2.1. *An arithmetic circuit C over a field $\mathbb{F}$ and the variable set $X = \{x_1, \ldots, x_n\}$ is a directed acyclic graph where each node is either a source node (indegree 0), or has indegree 2. The source nodes are labeled from the set $\mathbb{F} \cup X$, whereas the rest of the nodes are labeled $+$ or $\times$. There is a designated sink node (outdegree 0) called output gate.*

The source nodes are also called *input* gates. A circuit C computes a polynomial in a natural way. The input gates are labeled by either variables, or constants. Therefore, an input gate labeled by ℓ naturally computes the polynomial ℓ. A gate labeled by $+$ computes the sum of the polynomials computed by its children. A gate labeled by $\times$ computes the product of the polynomials computed by its children. The polynomial computed by C is the polynomial computed by the designated output gate. The *size* of an arithmetic circuit, denoted $\mathsf{size}(C)$, is the number of non-source nodes in the circuit. Sometimes the number of edges in the circuit is also considered as a measure of size, but note that the two measures are polynomially related. The *depth* of a circuit is the maximum length of a directed path from an input gate to the output gate. Sometimes it is useful to layer the vertices in C such that all edges in the underlying directed acyclic graph go from some layer i to $i + 1$. In such a case, we define the *width* of a circuit to be the maximum number of vertices in a layer.

If we restrict the general arithmetic circuit such that the outdegree of every node in the circuit is at most 1, we obtain a restricted model of arithmetic circuits called *formulas*. It is easy to observe that the graph underlying a formula is, in fact, a tree.

In the case of general arithmetic circuits, or formulas, the children of a $\times$ gate are unrestricted. We will study another model of algebraic computation, called *skew circuits*, in which multiplication gates are restricted such that at most one of the two children is a *non-input* gate. We now give an equivalent definition, but in terms of weighted graphs.

Definition 2.2.2. *An algebraic branching program (ABP) is a directed acyclic graph with two distinguished vertices, a designated source node s and a designated target node t. The edges are labeled by the elements in the field $\mathbb{F}$ or the set of variables $X = \{x_1, \ldots, x_n\}$.*

For any directed path ρ from s to t, the weight of ρ is the product of the labels of the edges on ρ. The polynomial computed by an ABP is the sum of weights of all paths from s to t.

The *size* of an algebraic branching program, denoted size, is the number of vertices in it. We will assume, without loss of generality, that algebraic branching programs are layered. That is, the vertices are partitioned into layers. The first layer contains only one vertex, the source node s. Similarly the last layer contains a single vertex, the target node t. Furthermore, all edges of the graph go from some layer i to $i + 1$. Again, analogous to circuits, the *width* of an algebraic branching program is defined to be the maximum number of vertices in a layer.

We say that a sequence of polynomials (f_n) is computable by a sequence of arithmetic circuits (or, branching programs) (C_n) if and only if C_n computes f_n for all n. Corresponding to two notions of computation we have two complexity measures, namely $\mathsf{size}_{\mathsf{c}}$ and $\mathsf{size}_{\mathsf{bp}}$. If $f = (f_n)$ is computable by a sequence of circuits (resp. branching programs) C_n, then $\mathsf{size}_{\mathsf{c}}(f) \leqslant \mathsf{size}(C_n)$ (resp. $\mathsf{size}_{\mathsf{bp}}(f) \leqslant \mathsf{size}(C_n)$). We now formalize the notion of feasible (easy to compute) families of polynomials.

Definition 2.2.3. *A sequence of polynomials (f_n) over $\mathbb{F}$ belongs to the class $\mathsf{VP}_{\mathbb{F}}$ if and only if (f_n) is a p-family, and is computable by a sequence of arithmetic circuits (C_n) over $\mathbb{F}$ such that $\mathsf{size}(C_n)$ is a p-bounded function of n.*

Definition 2.2.4. *A sequence of polynomials (f_n) over $\mathbb{F}$ belongs to the class $\mathsf{VF}_{\mathbb{F}}$ if and only if (f_n) is a p-family, and is computable by a sequence of formulas (C_n) over $\mathbb{F}$ such that $\mathsf{size}(C_n)$ is a p-bounded function of n.*

Definition 2.2.5. *A sequence of polynomials (f_n) over $\mathbb{F}$ belongs to the class $\mathsf{VBP}_{\mathbb{F}}$ if and only if (f_n) is a p-family, and is computable by a sequence of branching programs (B_n) over $\mathbb{F}$ such that $\mathsf{size}(B_n)$ is a p-bounded function of n.*

The following proposition is easy to show and well known (see, for instance, [MP08, Mah14]).

Proposition 2.2.6. *A sequence of polynomials* (f_n) *belongs to the class* VBP *iff it is computable by a sequence of skew circuits* (C_n) *such that* $\mathsf{size}(C_n)$ *is polynomially bounded.*

It is but natural to wonder: why not allow division gates in arithmetic circuits. However, over infinite fields, Strassen [Str73] showed that circuits with division gates can be simulated by circuits without division gates with only polynomial blow-up in size. (Hrubeš and Yehudayoff [HY11] extended this result to finite fields.)

We now introduce two polynomials of great significance in algebraic complexity theory, the *determinant polynomial* and the *permanent polynomial*. The family of determinant polynomials $\mathsf{Det} = (\mathsf{Det}_n)$ is such that the n-th polynomial is given by the determinant of an $n \times n$ symbolic matrix. We assume the entries are $\{x_{ij} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n\}$. Then

$$\mathsf{Det}_n := \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i=1}^{n} x_{i\sigma(i)} ,$$

where S_n is the group of permutations over n elements. Using Gaussian elimination we can construct a $\mathsf{poly}(n)$ sized arithmetic circuit using division gates that computes Det_n. Hence, from the discussion above, it follows that $\mathsf{Det} \in \mathsf{VP}_{\mathbb{F}}$ for all field $\mathbb{F}$. Det is also known to be in $\mathsf{VBP}_{\mathbb{F}}$ for all $\mathbb{F}$ (see [MV97] for a very elegant combinatorial proof.).

The permanent family $\mathsf{Perm} = (\mathsf{Perm}_n)$ of polynomials is a sequence where the n-th polynomial is the permanent of an $n \times n$ symbolic matrix. That is,

$$\mathsf{Perm}_n := \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)} .$$

Perm is *not* known to be in VP. In fact, it is not believed to be so. Valiant [Val79] defined an algebraic analog of the class NP and showed the permanent family to be complete for that class. VP can be thought of as an algebraic analog of the class P.

Definition 2.2.7. *A sequence of polynomials* (f_n) *over* $\mathbb{F}$ *belongs to the class* $\mathsf{VNP}_{\mathbb{F}}$ *if and* *only if there exists a polynomial p, and a sequence* $(g_n) \in \mathsf{VP}_{\mathbb{F}}$ *, such that for all n,*

$$f_n(\tilde{x}) = \sum_{\tilde{y} \in \{0,1\}^{p(n)}} g_n(\tilde{x}, \tilde{y}) \ .$$

The permanent family Perm is the canonical example of a family in VNP. There are many ways to see this. We will consider the definition of the permanent. It follows,

$$\mathsf{Perm}_n = \sum_{\substack{Y \in \{0,1\}^{n \times n} \\ Y \text{ is a permutation matrix}}} \prod_{i=1}^{n} \left(\sum_{j=1}^{n} Y_{ij} x_{ij} \right).$$

Now if we can write a small polynomial (i.e. in VP) that checks whether a given Boolean matrix is a permutation matrix or not, we would establish that Perm is in VNP. Recall that a permutation matrix is a $\{0, 1\}$-matrix such that each row and column contains exactly one 1. We construct indicator polynomials for the events, each row contains at most one 1, each column contains at most one 1, and each row contains at least one 1. The three events together implies that each row and column contains exactly one 1. So we get the following polynomial that, when evaluated on Boolean inputs, outputs 1 if the input is a permutation matrix, and 0 otherwise.

$$h_n(Y) = \prod_{i=1}^{n} \left(\prod_{\substack{j,k \in [n] \\ j \neq k}} (1 - Y_{ij} Y_{ik}) \right) \cdot \prod_{j=1}^{n} \left(\prod_{\substack{i,k \in [n] \\ i \neq k}} (1 - Y_{ij} Y_{kj}) \right) \cdot \prod_{i=1}^{n} \left(\sum_{j=1}^{n} Y_{ij} \right).$$

The first (resp. second) term in the product checks whether each row (resp. column) has at most one 1, and the third term checks whether each row has at least one 1. Therefore,

$$\mathsf{Perm}_n = \sum_{Y \in \{0,1\}^n} h_n(Y) \prod_{i=1}^{n} \left(\sum_{j=1}^{n} Y_{ij} x_{ij} \right).$$

Clearly, $h_n(Y) \prod_{i=1}^{n} \left(\sum_{j=1}^{n} Y_{ij} x_{ij} \right) \in \mathsf{VP}$. Hence, $\mathsf{Perm} \in \mathsf{VNP}$.

It follow from definitions that $\mathsf{VBP} \subseteq \mathsf{VP} \subseteq \mathsf{VNP}$. It is not known whether either of the containment is proper. Valiant's hypothesis says that the second containment is strict, that is, $\mathsf{VP} \subset \mathsf{VNP}$. In fact, Valiant gave evidence for his hypothesis. He described complete families of polynomials for these classes. These are families that in some sense capture the complexity of the class. Before we can talk about completeness, we must be able to compare two problems. Valiant [Val79] proposed a strict, but natural, notion of reduction between families of polynomials.

Definition 2.2.8. *A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is a* projection *of a polynomial $g \in \mathbb{F}[y_1, \ldots, y_m]$ if there exists a substitution map $\sigma\colon \{y_1, \ldots, y_m\} \to \mathbb{F} \cup \{x_1, \ldots, x_n\}$ such that*

$$f(x_1, \ldots, x_n) = g(\sigma(y_1), \ldots, \sigma(y_m)).$$

Further, a sequence of polynomials (f_n) is a p-projection *(or,* qp-projection*) of the family (g_n) if there exists a p-bounded (or, qp-bounded) function $t\colon \mathbb{N} \to \mathbb{N}$, such that for every n, f_n is a projection of $g_{t(n)}$.*

It is easily seen that all the classes VBP, VP and VNP are closed under p-projection. We can now formally define the notion of completeness.

Definition 2.2.9. *For an algebraic class C, a family of polynomials $f = (f_n)$ is said to be C-hard with respect to p-projections (or, qp-projection), if every family in C is a p-projection (or, qp-projection) of f. Furthermore, if $f \in \mathsf{C}$, it is said to be C-complete with respect to p-projections (or, qp-projections).*

For convenience, we will drop the explicit mention of the reduction if *hardness* is established under p-projections.

Valiant showed the permanent family Perm to be VNP-complete, and the determinant family Det to be VP-complete with respect to qp-projections.

Theorem 2.2.10 ([Val79])**.** *Over fields of characteristic other than 2, (Perm_n) is VNP-complete.*

Theorem 2.2.11 ([Val79]). *Over any field, the determinant family* (Det_n) *is* VP*-complete with respect to* qp*-projections.*

Note that the VP-hardness of Det is established with respect to qp-projections. It is not known whether Det is VP-hard with respect to p-projections. However, it is known since [Val79] that Det is VBP-hard with respect to p-projections. Later, it was also shown to be in VBP [Dam91, Tod92, Vin91, MV97]. Thus, the determinant family is complete for a possibly smaller class. No other natural family was known to be VP-complete with respect to p-projections. This lack of complete families of polynomials for the class VP will be the central theme to be explored in Chapter 3.

We end this section with a useful criterion, due to Valiant [Val79], to verify whether a polynomial family belongs to the class VNP. (See, also, [HWY10], or Section 2.3 in [Bür00a].)

Proposition 2.2.12 (Valiant's criterion, [Val79]). *Let* (f_n) *be a p-family of polynomials over any field. Suppose there exists a* $\#\mathsf{P}/\mathsf{poly}$ *algorithm which, given n and a monomial* $\mathfrak{m}$ *as inputs, compute the coefficient of* $\mathfrak{m}$ *in* f_n*. Then,* $(f_n) \in \mathsf{VNP}$.

Proof. Let $f_n(X_1, \ldots, X_t)$ be a polynomial of degree $D(n)$ over $t(n)$ variables, with coefficients expressible as $1 + 1 + \ldots + 1$ over the underlying field. Note that $D(n)$ and $t(n)$ are p-bounded in n. That is,

$$f_n = \sum_{\substack{\tilde{D}=(D_1,\ldots,D_t)\in\mathbb{N}^t \\ \sum_j D_j = D}} c(\tilde{D}) \left(\prod_{i=1}^{t} X_i^{D_i} \right),$$

where $c(\tilde{D})$ is the coefficient of the monomial given by the degree sequence $\tilde{D}$.

We give a short proof of the proposition following the arguments in [Bür00a]. For ease of presentation we will consider the case when algorithm computing the coefficients is in $\#\mathsf{P}$ rather than $\#\mathsf{P}/\mathsf{poly}$. Essentially the same argument goes through in the latter case too.

Fix $d := \lceil \log D \rceil$. Let $\phi\colon \{0,1\}^* \to \mathbb{N}$, in #P, be an algorithm that compute the coefficients of f_n given the degree sequence and n. We will denote a degree sequence by a $\{0,1\}$-matrix of size $t \times d$, where rows represent the individual degrees in binary. Then,

$$f_n = \sum_{E \in \{0,1\}^{t\times d}} \phi(E) \left(\prod_{i=1}^{t} X_i^{bin(E_i)} \right), \tag{2.1}$$

where E_i is the i-th row of the matrix E, and $bin(E_i)$ denotes the natural number given by the binary string E_i.

Since #3-SAT is #P-complete via parsimonious reductions, for every $n \in \mathbb{N}$, there exists a 3-CNF Φ_n with $\mathsf{poly}(n)$ clauses over the variable matrix E and some additional variables $Y_1, \ldots, Y_{m(n)}$, such that $m(n)$ is p-bounded in n, and for all $E \in \{0,1\}^{t\times d}$,

$$\phi(E) = \#\{\, Y \in \{0,1\}^{m(n)} \mid \Phi_n(E, Y) \text{ is true}\}.$$

We now arithmetize Φ_n, in an obvious way (see, for example, [Bür00a]), to obtain a polynomial p_n such that for all $E \in \{0,1\}^{t\times d}$,

$$\phi(E) = \sum_{Y \in \{0,1\}^{m(n)}} p_n(E, Y). \tag{2.2}$$

Observe that by construction (p_n) is in VP. We now define another polynomial h_n as follows:

$$h_n(X, E, Y) := p_n(E, Y) \prod_{i=1}^{t} \left(\prod_{j=1}^{d} \left(E_{i,j} X_i^{2^j} + 1 - E_{i,j} \right) \right).$$

From the definition of h_n, it is easily seen that $(h_n) \in$ VP. Furthermore, using Eq. (2.2), we have

$$\phi(E) \left(\prod_{i=1}^{t} X_i^{bin(E_i)} \right) = \sum_{Y \in \{0,1\}^{m(n)}} p_n(E, Y) \prod_{i=1}^{t} \left(\prod_{j=1}^{d} \left(E_{i,j} X_i^{2^j} + 1 - E_{i,j} \right) \right).$$

Thus, from Eq. (2.1), it now follows that

$$f_n = \sum_{E\in\{0,1\}^{t\times d}} \sum_{Y\in\{0,1\}^{m(n)}} p_n(E,Y) \prod_{i=1}^{t} \left(\prod_{j=1}^{d} \left(E_{i,j} X_i^{2^j} + 1 - E_{i,j} \right) \right) = \sum_{E\in\{0,1\}^{t\times d}} \sum_{Y\in\{0,1\}^{m(n)}} h_n(X,E,Y).$$

Since $(h_n) \in \mathsf{VP}$, we have therefore shown that $(f_n) \in \mathsf{VNP}$. □

2.3 Reductions

In this section, we will study the different kinds of reductions among families of polynomials. We have already seen *projections* (Definition 2.2.8) in the last section. It is a very strict notion of reduction, where a polynomial f is said to be a projection of g if f can be obtained from g by simply fixing some variables to constants and substituting the others with variables of f.

For example, let $g = y_1y_2 + y_3y_4$, then $x_1^2 + x_2^2$, $x_1 + x_2$, $x_1x_2 + 1$ and, 0 are some of the projections of g. But x_1^3, $x_1^2 - x_2^2$ and $x_1 + x_2 + 1$ are not projections of g.

A motivation to study such a restrictive kind of reduction is that it easily preserves/transfers computational hardness among problems. Indeed, if f is a projection of g, it is readily seen that g is *at least as hard as* f. In other words, an arithmetic circuit (or, ABP) for f can be obtained from a circuit (or, ABP) for g, via the substitution given by the projection with no increase in size. It also follows that projection is *transitive*. That is, if f is a projection of g and g is a projection of h, then f is a projection of h.

These properties also carry over to *p-projections*, and allow us to study a completeness theory for the algebraic classes. As noted earlier, the classes VBP, VP and VNP are closed under p-projections.

Furthermore, if the variables of g are allowed to be substituted by *linear polynomials* in the variables of f, such a projection is called *affine* projection. For instance, in the example above, we observed that $x_1^2 - x_2^2$ and $x_1 + x_2 + 1$ are not projections of $y_1y_2 + y_3y_4$. But they are projections when the substitutions are allowed to be affine linear functions of x_1 and x_2. However, x_1^3 is not a projection even when affine linear functions are allowed.

Bürgisser [Bür00a] observed that sometimes it is easier to establish that a polynomial f is a linear combination of polynomially many projections of g. This would be the case, for instance, if f is obtained via interpolation using g. Formally, he defined the reduction as follows.

Definition 2.3.1. *A family of polynomials (f_n) is a* linear p-projection *of a family (g_n) over $\mathbb{F}$ if there exists a p-bounded function $t\colon \mathbb{N} \to \mathbb{N}$, such that for all n, there are indices $i_1 \leqslant i_2 \leqslant \ldots \leqslant i_{t(n)} \leqslant t(n)$ and constants $\lambda_k \in \mathbb{F}$ such that f_n is a projection of the linear combination $\sum_{k=1}^{t(n)} \lambda_k g_{i_k}$. The sets of variables of g_{i_k}, for distinct k, are considered to be disjoint.*

Again it easily follows that the relation linear p-projection is transitive, and the classes $\mathsf{VBP}, \mathsf{VP},$ and VNP are closed under it. In general, it is not clear whether the linear combination $\sum_{k=1}^{t(n)} \lambda_k g_{i_k}$ is itself a projection of some $g_{m(n)}$, where m is a p-bounded function of n.

Definition 2.3.2. *A sequence of polynomials (f_n) is called* linearly closed *if and only if any linear combination $\sum_{k=1}^{t} \lambda_k f_{i_k}$ is a projection of some f_m, where m is a p-bounded function of the number of terms t and maximum index* $\max_k i_k$.

From VBP-completeness of the determinant, it follows that Det is linearly closed. Bürgisser [Bür00a] proved that the two notions of VNP-completeness, with respect to p-projections and linear p-projections, are in fact the same.

Proposition 2.3.3 ([Bür00a])**.** *Let $f = (f_n) \in \mathsf{VNP}$. Then f is VNP-complete with respect to p-projections if and only if (i) f is linearly closed, and (ii) f is VNP-complete with respect to linear p-projections.*

Over ordered fields, such as $\mathbb{Q}$ and $\mathbb{R}$, or, more generally, any totally ordered semi-ring, such as subring of $\mathbb{R}$, Boolean $\{\wedge, \vee\}$-semi-ring, or tropical semi-ring of real numbers under min and addition, we say that a projection is *monotone* if and only if all constants appearing in the substitution are *non-negative*.

Bürgisser [Bür99] introduced and studied the concept of oracle computations. He defined a reduction analogous to Turing reduction in the classical setting.

Definition 2.3.4. *We say that a sequence of polynomials $f = (f_n)$ is* c-reducible *to $g = (g_n)$ if there exists a circuit family (C_n) over the gates $+, \times$ and evaluations of g at previously computed values such that C_n computes f_n, and $\mathsf{size_c}(C_n)$ is a p-bounded function of n.*

Observe that, in c-reduction, evaluations of g are evaluations of some elements of the sequence g such that their indices are p-bounded in n. As before, we observe that c-reduction is transitive, and that the class VP is closed under c-reduction. The class VNP is also known to be closed under c-reduction, though it is not easy to establish (see [Poi08] for a proof).

The c-reduction is at least as powerful as p-projection, that is, if f is p-projection of g, then f c-reduces to g. In particular, the following implication always holds, f p-projection of $g \Rightarrow f$ linear p-projection of $g \Rightarrow f$ c-reduces to g. Moreover, the constant polynomial 0 is VP-complete with respect to c-reduction. Hence, c-reduction does not isolate algebraic classes lower than VP. To overcome this pitfall, we study a weaker variant of c-reduction. It can be regarded as an analogue of AC^0-Turing reductions in the Boolean world.

Definition 2.3.5. *We say that a sequence of polynomials $f = (f_n)$ is* constant-depth c-reducible *to $g = (g_n)$ if there exists a* constant-depth *circuit family (C_n) over the gates $+, \times$ and evaluations of g at previously computed values such that C_n computes f_n, and* $\mathsf{size_c}(C_n)$ *is a p-bounded function of n.*

2.4 Computation with symmetric matrices

Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial in n variables. A natural restriction in the study of the representation of polynomials by the determinant (or, permanent) of a matrix is to condition the matrix to be symmetric. That is, construct a symmetric matrix A with entries in $\mathbb{F} \cup \{x_1, \ldots, x_n\}$ such that $f = \mathsf{det}(A)$ (or, $f = \mathsf{perm}(A)$). Formally, we consider the following symmetric variants of the determinant and the permanent families. Let $X_n = [x_{i,j}]_{1 \leqslant i,j \leqslant n}$ be an $n \times n$ symmetric symbolic matrix, i.e., $x_{i,j} = x_{j,i}$. Then, $\mathsf{sym\text{-}Perm} = (\mathsf{sym\text{-}Perm}_n)$ is a family of polynomials where $\mathsf{sym\text{-}Perm}_n$ is the permanent of the matrix X_n. Similarly, we define the symmetric determinant family $\mathsf{sym\text{-}Det} = (\mathsf{sym\text{-}Det}_n)$.

Due to the preponderance of applications of the determinant, there has been a long line of work, from as early as the nineteenth century, on *symmetric determinantal representations* of polynomials [Hes55, Cay69, Dix02, Dic21, Cat81, Bea00, HMV06, HV07, Brä11, GKKP11, PSV11, NT12, Qua12, NPT13, Brä13, GMT13]. Some of these works, in fact, study symmetric determinantal representation with a further constraint that when all the variables are set to zero the matrix is positive semi-definite. Such representation is of significance in convex optimization.

In this section we study the representation of polynomials by the permanent of a symmetric matrix from the algebraic complexity point of view. Analogous question for the determinant was studied by Grenet et al. [GKKP11]. In particular, they showed Det is a p-projection of $\mathsf{sym\text{-}Det}$ over fields of char $\neq 2$. Thus, $\mathsf{sym\text{-}Det}$ is VBP-complete. Here we investigate $\mathsf{sym\text{-}Perm}$ from a similar viewpoint.

We start with some observations that easily follows from earlier work, for example, see [GKKP11]. For the remaining of this section we will fix the characteristic of the underlying field to be different from 2. Indeed, over fields of char 2, Grenet et al. [GMT13] showed that the polynomial $xy + z$ cannot be represented as the permanent of a symmetric matrix.

Proposition 2.4.1 (Universality)**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by an algebraic branching program of size s, and number of edges e. Then there exists an undirected graph G, with weights on edges from $\mathbb{F} \cup \{x_1, \ldots, x_n\}$, such that $f = \mathsf{perm}(A_G)$ where A_G is the weighted adjacency matrix of G. Moreover, the number of vertices in G is at most $(2(s-1)+1)$, and the number of edges is $(e+s)$.*

Proof. The proof follows from the proof of Theorem 5 in [GKKP11] along with an additional observation. It is easy to observe that the construction of the undirected graph given therein is such that the determinant and the permanent of the weighted adjacency matrix are equal. □

Clearly, $\mathsf{sym\text{-}Perm} \in \mathsf{VNP}$. Also, it follows from Proposition 2.4.1 that every polynomial family in VNP is a projection of $\mathsf{sym\text{-}Perm}$. But it is not clear whether this projection is a p-projection. We will now discuss this in detail, so let us state the question formally.

Question 2.4.1. *Over fields of characteristic different from 2, is $\mathsf{sym\text{-}Perm} = (\mathsf{sym\text{-}Perm}_n)$ VNP-hard with respect to p-projections?*

Given an $n \times n$ symbolic matrix X, consider the symmetric matrix $B := \begin{pmatrix} 0 & X \\ X^t & 0 \end{pmatrix}$ where X^t is the transpose of the matrix X. It immediately follows that $\mathsf{perm}(B) = \mathsf{perm}(X)^2 = \mathsf{Perm}_n^2$. However, Perm_n is an irreducible polynomial. This suggests an approach to compute Perm_n using factorization [Kal86, Kal87, Kal89, KT90] given oracle access to permanents of symmetric matrices.

Theorem 2.4.2 ([Bür00a])**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$, where $\mathbb{F}$ is a field of characteristic 0, be a polynomial of degree d. Then, there exists an arithmetic circuit which computes all the irreducible factors of f with $(d+1)^2$ evaluations of f and $\mathsf{poly}(n, d)$ arithmetic operations.*

Using the above theorem, it is easily seen that Perm_n c-reduces to $\mathsf{sym\text{-}Perm}_{2n}$. Hence, we obtain the following corollary.

Corollary 2.4.3. *Over fields of characteristic* 0*,* $\mathsf{sym\text{-}Perm}$ *is* VNP*-hard with respect to c-reductions.*

The reduction in the above hardness result can be improved to *constant-depth* c-reduction using the results of Oliveira [Oli15] on factoring of polynomials when the polynomial has low degree in each variable. For instance, here $\mathsf{perm}(B)$ has degree at most 2 in each variable.

In fact, if one goes through the proof of [Oli15] and, also, uses the fact $\mathsf{perm}(M)\mathsf{perm}(N) = \mathsf{perm}\begin{pmatrix} M & 0 \\ 0 & N \end{pmatrix}$, then Corollary 2.4.3 can be further improved to the following.

Corollary 2.4.4. *Over fields of characteristic* 0*,* $\mathsf{sym\text{-}Perm}$ *is* VNP*-hard with respect to linear p-projections.*

We now show a very neat and simple way to show that Perm_n can be written as a difference of two projections of $\mathsf{sym\text{-}Perm}_{10n}$. Furthermore, this way only requires char $\neq 2$.

Consider the polynomials $\mathsf{Perm}_n \pm \gamma$, where γ is a new and distinct variable that does not appear in the $n \times n$ symbolic matrix X_n.

Lemma 2.4.5. *There exists a* $5n \times 5n$ *matrix* Y_n *with entries from* $X_n \cup \{0, 1, -1, -1/2, 1/2, \gamma\}$ *such that* $\mathsf{perm}(Y_n) = \mathsf{Perm}_n + \gamma$*. Similarly, there exists a matrix* Z_n *such that* $\mathsf{perm}(Z_n) = \mathsf{Perm}_n - \gamma$*. In fact, the two matrices* Y_n *and* Z_n *differ in exactly one entry.*

Before we see the proof of Lemma 2.4.5, let us complete the argument that Perm_n is a projection of a linear combination of $\mathsf{sym\text{-}Perm}_{10n}$, with two summands.

Consider the following symmetric matrices,

$$B_n(\gamma) := \begin{pmatrix} 0 & Y_n \\ Y_n^t & 0 \end{pmatrix} \quad \text{and,} \quad B_n'(\gamma) := \begin{pmatrix} 0 & Z_n \\ Z_n^t & 0 \end{pmatrix}.$$

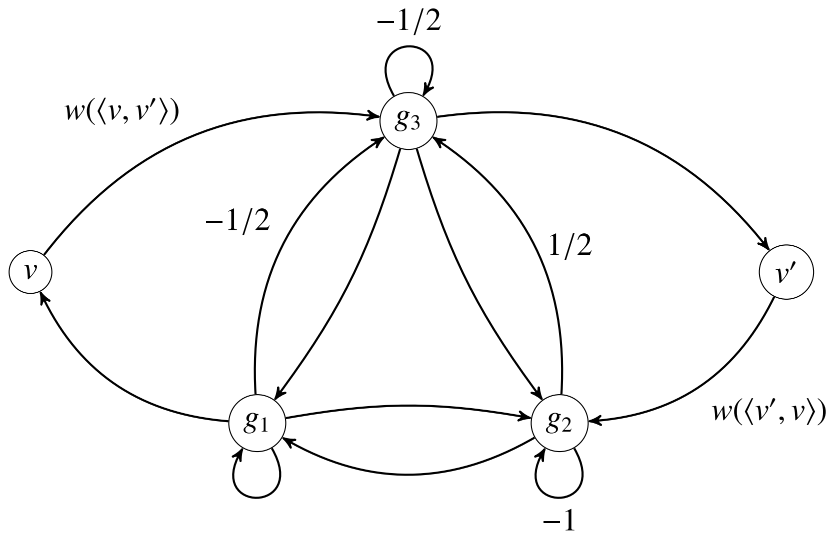

Figure 2.1: xor-gadget between parallel edges

Now using the algebraic identity, $(a+b)^2 - (a-b)^2 = 4ab$, it follows that $4 \cdot \gamma \cdot \mathsf{Perm}_n = \mathsf{perm}(B_n(\gamma)) - \mathsf{perm}(B'_n(\gamma))$. Thus, $\mathsf{Perm}_n = \mathsf{perm}(B_n(4^{-1})) - \mathsf{perm}(B'_n(4^{-1}))$. From the proof we see that we need 2^{-1} and 4^{-1} to exist. Hence, we obtain that, over fields of characteristic different from 2, Perm_n can be written as a difference of two projections of $\mathsf{sym\text{-}Perm}_{10n}$.

Theorem 2.4.6. *Over fields of characteristic not equal to 2,* sym-Perm *is* VNP*-complete with respect to linear p-projections. Furthermore, there are only two summands in the linear p-projections.*

Observe that bringing down the number of summands from 2 to 1 will solve the Question 2.4.1 in its entirety. We now proceed to give a proof of Lemma 2.4.5. For the ease of presentation we will identify matrices as directed edge-weighted graphs, and we will present the proof in a graph theoretic language.

Proof of Lemma 2.4.5. Let G_n be the complete directed edge-weighted graph on vertices $\{v_1, v_2, \ldots, v_n\}$ such that its weighted adjacency matrix is given by X_n. By $\mathsf{perm}(G_n)$ we will mean the permanent of the weighted adjacency matrix of G_n. Thus, $\mathsf{perm}(G_n) = \mathsf{Perm}_n$. A cycle cover of G_n is a set of edges such that together they form a disjoint union of simple cycles covering all the vertices of G_n. It is well known and easy to show that

$$\mathsf{perm}(G_n) = \sum_{C \in \mathrm{CC}(G_n)} \prod_{e \in C} w(e),$$

where $\mathrm{CC}(G_n)$ is the set of all cycle covers of G_n, and $w(e)$ is the weight of the edge e.

To establish the Lemma it suffices to construct a graph H_n on $5n$ vertices such that $\mathsf{perm}(H_n) = \mathsf{perm}(G_n) + \gamma$. (The construction for $\mathsf{Perm}_n - \gamma$ is similar.)

Consider a directed (simple) cycle $v'_1 \to v'_2 \to \cdots \to v'_n \to v'_1$ on n vertices. Only the edges present in the cycle have non-zero weight and they are all equal to 1. We will denote it by C_n. Now consider the disjoint union of G_n and C_n. To this graph we add the following directed edges $\{\langle v_i, v'_i\rangle, \langle v'_i, v_i\rangle \mid 1 \leqslant i \leqslant n\}$. The weights are given as follows: $w(\langle v_1, v'_1\rangle) = \gamma$, $w(\langle v'_1, v_1\rangle) = 1$, and for all $i \neq 1$, $w(\langle v_i, v'_i\rangle) = w(\langle v'_i, v_i\rangle) = 1$. Further add an *xor-gadget* between each pair of parallel edges $\{\langle v_i, v'_i\rangle, \langle v'_i, v_i\rangle\}$, with three new vertices g_{i1}, g_{i2}, and g_{i3}, as shown in Fig. 2.1. (The present edges, with unspecified weights, have weights equal to 1.) We call this graph H_n. Clearly, H_n has $5n$ vertices.

An *xor-gadget* [vzG87, Bür00a] is the complete directed graph on 3 vertices such that the weights on the edges are given by the matrix K (cf. Fig. 2.1),

$$K := \begin{pmatrix} 1 & 1 & -\frac{1}{2} \\ 1 & -1 & \frac{1}{2} \\ 1 & 1 & -\frac{1}{2} \end{pmatrix}.$$

Let $T_i := \left\{\langle v_i, g_{i3}\rangle, \langle g_{i3}, v'_i\rangle, \langle v'_i, g_{i2}\rangle, \langle g_{i1}, v_i\rangle\right\}$, and $T := \bigcup_{i=1}^{n} T_i$. The significance of the xor-gadget follows from the next claim.

Claim 2.4.1. *Fix an $i \in [n]$. Then, the permanent of H_n equals the sum of the weights of all cycle covers of H_n that either contain all the edges in T_i, or none.*

Proof. For a set $S \subseteq T_i$, we define $\mathrm{CC}_{H_n}(S)$ to be the set of all cycle covers of H_n that, among the edges in T_i, contain exactly the edges in S. From inspection, it follows that the following sets are the only possibilities for non-empty $\mathrm{CC}_{H_n}(S)$:

$$\varnothing, T_i, \{\langle v_i, g_{i3}\rangle, \langle g_{i3}, v_i'\rangle\}, \{\langle v_i', g_{i2}\rangle, \langle g_{i1}, v_i\rangle\}, \{\langle v_i, g_{i3}\rangle, \langle g_{i1}, v_i\rangle\}, \{\langle v_i', g_{i2}\rangle, \langle g_{i3}, v_i'\rangle\}.$$

The set of all cycle covers of H_n is partitioned into the six sets based on which subset of T_i they contain. The claim follows if we show that for $S \notin \{\varnothing, T_i\}$, the sum of the weights of all cycle covers in $\mathrm{CC}_{H_n}(S)$ is zero. Let $K[R \mid C]$ denote the minor of K obtained after removing rows in R and columns in C.

- If $S = \left\{\langle v_i, g_{i3}\rangle, \langle g_{i3}, v_i'\rangle\right\}$, the contribution of $\mathrm{CC}_{H_n}(S)$ to the permanent of H_n is 0 because $\mathsf{perm}(K[3 \mid 3]) = 0$.
- If $S = \left\{\langle v_i', g_{i2}\rangle, \langle g_{i1}, v_i\rangle\right\}$, the contribution is 0 because $\mathsf{perm}(K[1 \mid 2]) = 0$.
- If $S = \{\langle v_i, g_{i3}\rangle, \langle g_{i1}, v_i\rangle\}$, the contribution is 0 because $\mathsf{perm}(K[1 \mid 3]) = 0$.
- If $S = \left\{\langle v_i', g_{i2}\rangle, \langle g_{i3}, v_i'\rangle\right\}$, the contribution is 0 because $\mathsf{perm}(K[3 \mid 2]) = 0$.

□

Furthermore, if a cycle cover of H_n contains some T_i, then for the possibility of it contributing positively to the permanent of H_n, it must contain all of T. It follows by using Claim 2.4.1 with a further observation that the set of vertices v_j', such that T_j is disjoint from the cycle cover, must be covered among themselves by a cycle using edges of C_n. But this is an impossibility since for some i, v_i' is covered by the edges from T_i. Therefore, there are two types of cycle covers of H_n: (*i*) a disjoint union of a cycle cover of G_n, a cycle cover of xor-gadgets, and C_n, and (*ii*) made up of "parallel" edges (i.e., contains all of T). The contribution of cycle covers from Case (*i*) is $\mathsf{perm}(G_n)$, since $\mathsf{perm}(K) = 1$ and weight of C_n equals 1. In Case (*ii*), since there is a unique cycle cover that contains T, we get a contribution of γ. Thus, we obtain $\mathsf{perm}(H_n) = \mathsf{perm}(G_n) + \gamma = \mathsf{Perm}_n + \gamma$.

$\square$

2.5 Monotone projection and lower bounds

In this section, we will prove lower bounds with respect to monotone projections. Recall that the definition of monotone projection is valid only over totally ordered semi-rings.

In general, our endeavour to prove non-trivial lower bounds have met with failure. Nevertheless, we have made considerable progress in restricted settings, such as unconditional lower bounds against monotone computations [SS77, SS80, Raz85b, Raz85a, AB87, JS82, TT94]. In particular, Razborov [Raz85a] proved that computing the permanent, over the Boolean $\{\wedge, \vee\}$-semi-ring, requires monotone circuits of size at least $n^{\Omega(\log n)}$. Till date, this is the best lower bound known over the Boolean $\{\wedge, \vee\}$-semi-ring. Jukna [Juk14] observed that if the Hamiltonian cycle polynomial family is a *monotone* p-projection of the permanent family, over the Boolean $\{\wedge, \vee\}$-semi-ring, then one would get a lower bound of $2^{n^{\Omega(1)}}$ for monotone circuits computing the permanent, thus improving on [Raz85a].

Let us pause for a moment to recall the definitions of the polynomials of our interest. Given an $n \times n$ symbolic matrix, we have,

$$\mathsf{Perm}_n := \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}, \ \mathsf{HC}_n := \sum_{\substack{\sigma \in S_n \\ \sigma \text{ is a } n\text{-cycle}}} \prod_{i=1}^{n} x_{i,\sigma(i)}, \ \text{ and } \ \mathsf{Clique}_n^k := \sum_{\substack{S \subseteq [n] \\ |S|=k}} \prod_{\substack{i,j \in S \\ i<j}} x_{i,j}.$$

Observe that Clique_n^k is defined over the complete undirected graph on n vertices, whereas Perm_n and HC_n are defined over the complete directed graph on n vertices. We fix $k = \sqrt{n}$ to obtain a specific family of clique polynomials $\mathsf{Clique}^{\sqrt{n}} = (\mathsf{Clique}_n^{\sqrt{n}})$. The *Hamiltonian cycle* family is given by $\mathsf{HC} = (\mathsf{HC}_n)$.

It is known [Val79] that $\mathsf{Clique}^{\sqrt{n}}$ is a monotone p-projection of HC. In fact, $\mathsf{Clique}_n^{\sqrt{n}}$ is a monotone projection of HC_{25n^2} [AB87]. Now if it were the case that HC is a monotone

p-projection of Perm, then by transitivity, $\mathsf{Clique}^{\sqrt{n}}$ would be a monotone p-projection of Perm. Then, using the $2^{n^{\Omega(1)}}$ lower bound of Alon and Boppana [AB87] for $\mathsf{Clique}_n^{\sqrt{n}}$, we would get a lower bound of $2^{n^{\Omega(1)}}$ for Perm_n.

The importance of Jukna's observation is also highlighted by the fact that such a monotone p-projection, over the reals, would give an alternate proof of the fact that computing permanent by monotone circuits over $\mathbb{R}$ requires size at least $2^{n^{\Omega(1)}}$. Jerrum and Snir [JS82] proved that the permanent requires monotone circuits of size $2^{\Omega(n)}$ over $\mathbb{R}$ and tropical semi-ring.

Grochow, in [Gro15], resolved the question, whether HC is a monotone p-projection of Perm, in negative. By establishing a connection between monotone projections and extended formulations of linear programs, he showed that the Hamiltonian cycle polynomial is not a monotone *sub-exponential-size* projection of the permanent.

This answered Jukna's specific question about the Hamiltonian cycle in its entirety, but the underlying motivation still remains unanswered. That is, *Is* $\mathsf{Clique}^{\sqrt{n}}$ *a monotone p-projection of* Perm*?* May be not via the Hamiltonian cycle polynomial, but, say, via the *satisfiability* polynomial [Val79]. It is known (see Section 5 [AB87]) that $\mathsf{Clique}_n^{\sqrt{n}}$ is a monotone projection of the satisfiability polynomial over $O(n^4)$ variables. Here we answer the main motivation by proving that $\mathsf{Clique}^{\sqrt{n}}$ is *not* a monotone p-projection of Perm. In fact, if $\mathsf{Clique}_n^{\sqrt{n}}$ is a monotone projection of $\mathsf{Perm}_{t(n)}$, then $t(n)$ must be at least $2^{\Omega(\sqrt{n})}$. Our proof technique is the same as Grochow [Gro15]. We further use the proof idea to establish that some explicit non-negative polynomials (i.e. coefficients are non-negative) are not monotone p-projection of Perm.

Before proceeding further, we set up the tools required for the proof.

2.5.1 Preliminaries

For a set of vectors $S = \{v_1, \ldots, v_m\} \subseteq \mathbb{R}^n$, we denote the convex hull of the set S by conv S. In other words,

$$\text{conv } S := \left\{ \sum_{i=1}^{m} \alpha_i v_i \mid \alpha_i \geqslant 0, 1 \leqslant i \leqslant m, \text{ and } \sum_{i=1}^{m} \alpha_i = 1 \right\}.$$

For any polynomial p in n variables, let $\mathsf{Newt}(p)$ denote the polytope in $\mathbb{R}^n$ that is convex hull of the vectors of exponents of monomials of p. For example, consider $g(y_1, y_2, y_3, y_4) = y_1y_2 + y_3y_4$, then $\mathsf{Newt}(g) = \text{conv }\{(\,1\;1\;0\;0\,)^{\mathrm{t}}, (\,0\;0\;1\;1\,)^{\mathrm{t}}\}$.

The *correlation polytope* $\mathsf{COR}(n)$ is defined as the convex hull of $n \times n$ binary symmetric matrices of rank 1. That is, $\mathsf{COR}(n) := \text{conv }\{vv^{\mathrm{t}} \mid v \in \{0, 1\}^n\}$. For any Boolean formula ϕ on n variables, let $\mathsf{p\text{-}SAT}(\phi)$ denote the polytope in $\mathbb{R}^n$ that is the convex hull of all satisfying assignments of ϕ, i.e. conv $\{x \in \{0, 1\}^n \mid \phi(x) = 1\}$. Let $K_n = (V_n, E_n)$ denote the n-vertex complete graph. The travelling salesperson (TSP) polytope is defined as the convex hull of the characteristic vectors of all subsets of E_n that define a Hamiltonian cycle in K_n.

For a polytope P, let $\mathsf{c}(P)$ denote the minimal number of linear inequalities needed to define P. A polytope $Q \subseteq \mathbb{R}^m$ is an *extension* of $P \subseteq \mathbb{R}^n$ if there is a linear map $\pi\colon \mathbb{R}^m \to \mathbb{R}^n$ such that $\pi(Q) = P$. The *extension complexity* of P, denoted $\mathsf{xc}(P)$, is the minimum size $c(Q)$ of any extension Q (of any dimension) of P.

The following are straightforward, see for instance [Gro15, FMP⁺15].

Fact 2.5.1. *1.* $\mathsf{c}(\mathsf{Newt}(\mathsf{Perm}_n)) \leqslant 2n$.

2. *If polytope Q is an extension of polytope P, then $xc(P) \leqslant \mathsf{xc}(Q)$.*

We use the following recent results.

Lemma 2.5.2 ([Gro15])**.** *Let $f(x_1, \ldots, x_n)$ and $g(y_1, \ldots, y_m)$ be polynomials over a totally* *ordered semi-ring R, with non-negative coefficients. If f is a monotone projection of g, then the intersection of* $\mathsf{Newt}(g)$ *with some linear subspace is an extension of* $\mathsf{Newt}(f)$*. In particular,* $\mathsf{xc}(\mathsf{Newt}(f)) \leqslant m + \mathsf{c}(\mathsf{Newt}(g))$.

Theorem 2.5.3 ([FMP+15])**.** *There exists some constant* $C > 0$ *such that for all n,* $\mathsf{xc}(\mathsf{COR}(n)) \geqslant 2^{Cn}$.

Theorem 2.5.4 ([AT13])**.** *For every n, there exists a 3SAT formula* ϕ *with* $O(n)$ *variables and* $O(n)$ *clauses such that* $\mathsf{xc}(\mathsf{p\text{-}SAT}(\phi)) \geqslant 2^{\Omega(\sqrt{n})}$.

Theorem 2.5.5 ([Rot14])**.** *The extension complexity of the TSP polytope is* $2^{\Omega(n)}$.

2.5.2 The Clique polynomial

In this subsection, we will show that $\mathsf{Clique}^{\sqrt{n}}$ is not a monotone p-projection of Perm. To establish this we will consider a different polynomial $\mathsf{Clique}^* = (\mathsf{Clique}_n^*)$ that counts all cliques in a graph. Recall, $\mathsf{Clique}_n^{\sqrt{n}}$ enumerates only $\sqrt{n}$-sized cliques. More formally,

$$\mathsf{Clique}_n^* := \sum_{S \subseteq [n]} \prod_{i \in S} x_{i,i} \prod_{\substack{i,j \in S \\ i<j}} x_{i,j}.$$

We first show that proving monotone projection lower bound against Clique^* suffices to establish lower bound against $\mathsf{Clique}^{\sqrt{n}}$. The proof is basically the VNP-completeness proof of $\mathsf{Clique}_n^{n/2}$ (see [Hru15]).

Lemma 2.5.6. *The family* Clique^* *is a monotone p-projection of the family* $\mathsf{Clique}^{\sqrt{n}}$*. In particular,* Clique_n^* *is a monotone projection of* $\mathsf{Clique}_{(n+1)^2}^{n+1}$.

Proof. In fact, we will show that Clique_n^* is a monotone projection of $\mathsf{Clique}_{2n+1}^{n+1}$. Then we add dummy vertices to establish the lemma.

Let G_n be a complete undirected graph on n vertices $\{v_1, \ldots, v_n\}$ with edge weights $x_{i,j}$ on the edge (v_i, v_j). Let G'_n be a complete undirected graph on the vertex set $\{v'_1, \ldots, v'_n\}$ with every edge having weight 1. We also add the following set $\{(v_i, v'_j) \mid i \neq j\}$ of cross edges between G_n and G'_n. The edges in this set also have weight 1. To this graph we add a new vertex u such that it is adjacent to every vertex in $G_n \cup G'_n$. The edges adjacent to vertices in G'_n have weight 1. For the vertices in G_n the weight of the edge (u, v_i) is $x_{i,i}$. We call this graph, on $2n + 1$ vertices, H_n. We claim that there is a one-to-one correspondence between cliques in G_n (of all sizes) and $(n + 1)$-sized cliques in H_n. Let $S \subseteq \{v_1, \ldots, v_n\}$ be a subset of vertices such that they form a clique in G_n. Consider the following map which is easily seen to be bijective. Map S to the clique on the following set of vertices in H_n: $S \cup \{v'_j \mid j \notin S\} \cup \{u\}$. Thus, H_n gives a projection of $\mathsf{Clique}_{2n+1}^{n+1}$ that equals Clique_n^*. Since 0 and 1 are the only constants used in the projection, it is also a monotone projection.

To obtain the lemma we add n^2 isolated vertices to H_n. □

Theorem 2.5.7. *Over the reals (or any totally ordered semi-ring), the family* Clique^* *is not a monotone p-projection of the* Perm *family. In fact, if* Clique_n^* *is a monotone projection of* $\mathsf{Perm}_{t(n)}$*, then* $t(n) \geqslant 2^{\Omega(n)}$.

Proof. Let Q be the Newton polytope of Clique_n^*. It resides in $N := \binom{n}{2} + n$ dimensions. Furthermore, it is the convex hull of vectors of the form $\langle \tilde{a}, \tilde{b} \rangle$ where $\tilde{a} \in \{0, 1\}^{\binom{n}{2}}$ is the characteristic vector of the set of edges of the clique over the set of vertices given by $\tilde{b} \in \{0, 1\}^n$, in the complete undirected graph K_n. We will index a vector in N dimensions by pairs (i, j) such that $1 \leqslant i \leqslant j \leqslant n$.

Let us now consider the linear map $\ell \colon \mathbb{R}^N \to \mathbb{R}^{n \times n}$, defined as $\ell(A) := B$, where for $1 \leqslant i \leqslant j \leqslant n$, $B_{i,j} = B_{j,i} = A_{(i,j)}$. We now claim that under the map ℓ, Q is mapped to the correlation polytope $\mathsf{COR}(n)$. It suffices to show that vertices of Q under the map ℓ are mapped into $\mathsf{COR}(n)$, and every vertex of $\mathsf{COR}(n)$ has a pre-image in Q under ℓ. Indeed ℓ maps the vertices of Q to the vertices of $\mathsf{COR}(n)$ bijectively. It follows from the map that a vertex $\langle \tilde{a}, \tilde{b} \rangle$ of Q is mapped to the vertex $\tilde{b}\tilde{b}^{\mathrm{t}}$ of $\mathsf{COR}(n)$. Furthermore, the pre-image of a vertex $\tilde{b}\tilde{b}^{\mathrm{t}}$ of $\mathsf{COR}(n)$ is the clique given by the upper-triangular and diagonal entries of $\tilde{b}\tilde{b}^{\mathrm{t}}$. Thus Q is an extension of $\mathsf{COR}(n)$, so by Fact 2.5.1 (2), $\mathsf{xc}(\mathsf{COR}(n)) \leqslant \mathsf{xc}(Q)$.

Suppose Clique^*_n is a monotone projection of $\mathsf{Perm}_{t(n)}$. By Fact 2.5.1 (1) and Lemma 2.5.2, $\mathsf{xc}(\mathsf{Newt}(\mathsf{Clique}^*_n)) = \mathsf{xc}(Q) \leqslant t(n)^2 + c(\mathsf{Perm}_{t(n)}) \leqslant O(t(n)^2)$. From the preceding discussion and Theorem 2.5.3, we get $2^{\Omega(n)} \leqslant \mathsf{xc}(\mathsf{COR}(n)) \leqslant \mathsf{xc}(Q) \leqslant O(t(n)^2)$. Therefore, it follows that $t(n)$ is at least $2^{\Omega(n)}$. □

Theorem 2.5.8. *Over the reals (or any totally ordered semi-ring), the family* $\mathsf{Clique}^{\sqrt{n}}$ *is not a monotone p-projection of the* Perm *family. In fact, if* $\mathsf{Clique}^{\sqrt{n}}_n$ *is a monotone projection of* $\mathsf{Perm}_{t(n)}$, *then* $t(n) \geqslant 2^{\Omega(\sqrt{n})}$.

Proof. Suppose $\mathsf{Clique}^{\sqrt{n}}_n$ is a monotone projection of $\mathsf{Perm}_{t(n)}$. From Lemma 2.5.6, it follows that Clique^*_n is a monotone projection of $\mathsf{Perm}_{t((n+1)^2)}$. Hence, from Theorem 2.5.7 we get $t(n^2) \geqslant 2^{\Omega(n)}$. Thus, $t(n) \geqslant 2^{\Omega(\sqrt{n})}$. □

Remark 2.5.1. *It is easily seen that if a polynomial f over n-variables is an affine projection of* Perm_m, *then f is a (simple) projection of* $\mathsf{Perm}_{m(n+1)}$. *Hence, Theorem 2.5.7 and Theorem 2.5.8 holds even when we consider monotone affine projections of the permanent.*

2.5.3 Other polynomials

We now consider the intermediate polynomial families, $\mathsf{Sat}^{\mathsf{q}} = (\mathsf{Sat}^{\mathsf{q}}{}_n)$ and $\mathsf{Clow}^{\mathsf{q}} = (\mathsf{Clow}^{\mathsf{q}}{}_n)$. Recall from Section 1.1, the above two polynomials are shown to be VNP-intermediate, in Chapter 4, when considered over the field $\mathbb{F}_q$. In this subsection, we consider these non-negative polynomials over $\mathbb{R}$ (or, any totally ordered semi-ring), and show lower bounds against monotone projections from the Perm to $\mathsf{Sat}^{\mathsf{q}}$ and $\mathsf{Clow}^{\mathsf{q}}$. We recall the definitions first.

The *satisfiability* polynomial $\mathsf{Sat}^{\mathsf{q}} = (\mathsf{Sat}^{\mathsf{q}}{}_n)$: For each n, let Cl_n denote the set of all possible clauses of size 3 over $2n$ literals. There are n variables $\tilde{X} = \{X_i\}_{i=1}^n$, and also $8n^3$

clause-variables $\tilde{Y} = \{Y_c\}_{c\in\mathsf{Cl}_n}$, one for each 3-clause c.

$$\mathsf{Sat^q}_n := \sum_{a\in\{0,1\}^n} \left(\prod_{i=1}^{n} X_i^{a_i(q-1)} \right) \left(\prod_{\substack{c\ \in\mathsf{Cl}_n \\ a \text{ satisfies } c}} Y_c^{q-1} \right).$$

The *clow* polynomial $\mathsf{Clow^q} = (\mathsf{Clow^q}_n)$: A clow in an n-vertex graph is a closed walk of length exactly n, in which the minimum numbered vertex (called the head) appears exactly once.

$$\mathsf{Clow^q}_n := \sum_{w:\text{ clow of length } n} \left(\prod_{e:\text{ edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v:\text{ vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

(If an edge e is used k times in a clow, it contributes $X_e^{k(q-1)}$ to the monomial.)

Theorem 2.5.9. *Over the reals (or any totally ordered semi-ring), for any integer $q \geqslant 2$, the families $\mathsf{Sat^q}$ and $\mathsf{Clow^q}$ are not monotone affine p-projections of the Perm family. In particular, if $\mathsf{Sat^q}_n$ (resp. $\mathsf{Clow^q}_n$) is a monotone affine projection of $\mathsf{Perm}_{t(n)}$, then $t(n)$ is at least $2^{\Omega(\sqrt{n})}$ (resp. $2^{\Omega(n)}$).*

Proof. Let ϕ be a 3SAT formula with n variables and m clauses as given by Theorem 2.5.4. For the polytope $P = \mathsf{p\text{-}SAT}(\phi)$, $\mathsf{xc}(P)$ is high.

Let Q be the Newton polytope of $\mathsf{Sat^q}_n$. It resides in N dimensions, where $N = n + |\mathsf{Cl}_n| = n + 8n^3$, and is the convex hull of vectors of the form $(q-1)\langle\tilde{a}\tilde{b}\rangle$ where $\tilde{a} \in \{0,1\}^n$, $\tilde{b} \in \{0,1\}^{N-n}$, and for all $c \in \mathsf{Cl}_n$, $\tilde{a}$ satisfies c iff $b_c = 1$. For each $\tilde{a} \in \{0,1\}^n$, there is a unique $\tilde{b} \in \{0,1\}^{N-n}$ such that $(q-1)\langle\tilde{a}\tilde{b}\rangle$ is in Q.

Define the polytope R, also in N dimensions, to be the convex hull of vectors that are vertices of Q and also satisfy the constraint $\sum_{c\in\phi} b_c \geqslant m$. This constraint discards vertices of Q where $\tilde{a}$ does not satisfy ϕ. Thus R is an extension of P (projecting the first n coordinates of points in R gives a $(q-1)$-scaled version of P), so by Fact 2.5.1 (2),

$\mathsf{xc}(P) \leqslant \mathsf{xc}(R)$. Further, we can obtain an extension of R from any extension of Q by adding just one inequality; hence $\mathsf{xc}(R) \leqslant 1 + \mathsf{xc}(Q)$.

Suppose $\mathsf{Sat^q}_n$ is a monotone affine projection of $\mathsf{Perm}_{t(n)}$. By Fact 2.5.1 (1) and Lemma 2.5.2, $\mathsf{xc}(\mathsf{Newt}(\mathsf{Sat^q})) = \mathsf{xc}(Q) \leqslant t(n) + c(\mathsf{Perm}_{t(n)}) \leqslant O(t(n))$. From the preceding discussion and By Theorem 2.5.4, we get $2^{\Omega(\sqrt{n})} \leqslant \mathsf{xc}(P) \leqslant \mathsf{xc}(R) \leqslant 1 + \mathsf{xc}(Q) \leqslant O(t(n))$. It follows that $t(n)$ is at least $2^{\Omega(\sqrt{n})}$.

For the $\mathsf{Clow^q}$ polynomial, let P be the TSP polytope and Q be $\mathsf{Newt}(\mathsf{Clow^q})$. The vertices of Q are of the form $(q-1)\tilde{a}\tilde{b}$ where $\tilde{a} \in \{0,1\}^{\binom{n}{2}}$ picks a subset of edges, $\tilde{b} \in \{0,1\}^n$ picks a subset of vertices, and the picked edges form a length-n clow touching exactly the picked vertices. Define polytope R by discarding vertices of Q where $\sum_{i\in[n]} b_i < n$. Now, using Theorem 2.5.5, the same argument as above works. $\square$

2.6 Closure properties

We begin by recalling the definition (Definition 2.2.7) of VNP. A polynomial family (f_n) is said to be in VNP if and only if there exist a family $(g_n) \in \mathsf{VP}$ such that for all n,

$$f_n(x_1, \ldots, x_{q(n)}) = \sum_{\bar{y}\in\{0,1\}^{p(n)}} g_n(x_1, \ldots, x_{q(n)}, y_1, \ldots, y_{p(n)}), \tag{2.3}$$

for some polynomial functions $p(n)$ and $q(n)$. We define *exponential sum* of a polynomial g with respect to a variable set S to be the sum of all $2^{|S|}$ projections of g, where a projection is obtained by setting the variables in S to $\{0,1\}$-value (cf. Eq. (2.3)). Hence, alternatively, one can think of VNP as *exponential sums* of polynomial sized circuits, denoted $\mathsf{VNP} = \sum \cdot \mathsf{VP}$. Consequently the VP versus VNP question can be reformulated as understanding the power of exponential sums.

In the foundational paper [Val82], Valiant observed that exponential sums of polynomial sized formulas ($\sum \cdot \mathsf{VF}$) or ($\sum \cdot \mathsf{VP}_e$) exactly capture exponential sums of polynomial sized circuits ($\sum \cdot \mathsf{VP}$). That is, $\mathsf{VNP}_e = \mathsf{VNP}$ (see also [MP08]). He used this observation crucially to show that the *permanent* polynomial is VNP-hard. Therefore, from Valiant's observation, it follows that $\sum \cdot \mathsf{VF} = \sum \cdot \mathsf{VBP} = \sum \cdot \mathsf{VP} = \mathsf{VNP}$.

Valiant's observation raises a natural question to study: How powerful are *exponential sums* of *restricted* circuit classes?

A natural restriction on arithmetic circuits is *multilinearity*. A polynomial is called *multilinear* if each variable in the polynomial has degree at most 1. An arithmetic circuit is called *multilinear* if every gate in it computes a multilinear polynomial. Furthermore, if for every product gate, the sub-circuits rooted at the left and right child are variable-disjoint, then the circuit is called *syntactic multilinear*.

The exponential summation under the restriction of syntactic multilinearity was studied by Jansen et al. [MR08, JR09, JMR13]. They showed that syntactic multilinear classes are closed under exponential sums. In particular, exponential summation does not add any power to syntactic multilinear formulas. Contrast this with the case of general formulas, where it become as powerful as VNP. Exponential summations of polynomials were also studied by Juma et al. [JKRS09]. Their motivation was to obtain query algorithms for #SAT that are better than brute-force. They proved that over fields of characteristic different from 2, multilinear polynomials are closed under exponential sums (Observation 1.3, [JKRS09]).

Here we study the exponential summation under the restriction of *multilinearity* (not necessarily syntactic). Using techniques different from those used in [JMR13, JKRS09], we extend their results by showing that over any field, exponential summation does not add power to multilinear circuit classes. We obtain our result (Theorem 2.6.1) by considering summations of general polynomials, but the summation is over variables that have degree at most 1 in the polynomial. It is shown that such a summation over multilinear variables is as good as evaluating the polynomial at one or a small number of points.

Theorem 2.6.1. *Let $f(x_1, \ldots, x_N, y_1, \ldots, y_m)$ be a polynomial that is multilinear in the $Y = \{y_1, \ldots, y_m\}$ variables. Let $h(X)$ be the exponential sum polynomial*

$$h(X) = \sum_{e \in \{0,1\}^m} f(X, e_1, \ldots, e_m).$$

If f has an efficient computation, so does h. The following table gives upper bounds on the complexity measures of h in terms of the corresponding measures of f.

		Char $\neq 2$	*Char* $= 2$	
			infinite fields	*finite fields*
Circuit (size,width)	f	s, w	s, w	s, w
	h	$s+1, w$	$3s(m+1), w+1$	$s(m+1)^2, w(m+1)$
ABP (size,width)	f	s, w	s, w	s, w
	h	$s+1, w$	$3s(m+1), w+2$	$s(m+1), w(m+1)$
Formula size	f	s	s	s
	h	$s+1$	$O(s)$ [JMR13]	$O(s)$ [JMR13]

Furthermore, if the circuit/ABP/formula for f is multilinear, then so is the circuit/ABP/formula for h.

Proof. Let $f(x_1, \ldots, x_N, y_1, \ldots, y_m)$ be some polynomial that is multilinear in the $Y = \{y_1, \ldots, y_m\}$ variables. Then there are polynomials $f_S(X)$ in the variables $X = \{x_1, \ldots, x_N\}$, one for each $S \subseteq [m]$, such that we can express f in terms of them:

$$f(X, Y) = \sum_{S \subseteq [m]} f_S(X) \prod_{i \in S} y_i.$$

Now consider the exponential Boolean sum

$$\begin{aligned} h(X) &= \sum_{e \in \{0,1\}^m} f(X, e) = \sum_{e \in \{0,1\}^m} \sum_{S \subseteq [m]} f_S(X) \prod_{i \in S} e_i \\ &= \sum_{e \in \{0,1\}^m} \sum_{S \subseteq [m] : i \in S \Rightarrow e_i = 1} f_S(X) \end{aligned}$$

$$= \sum_{S \subseteq [m]} f_S(X) \sum_{e \in \{0,1\}^m : i \in S \Rightarrow e_i = 1} 1$$
$$= \sum_{S \subseteq [m]} f_S(X) 2^{m-|S|}.$$

Char $\neq 2$ If the field has characteristic other than 2, then

$$h(X) = 2^m \sum_{S \subseteq [m]} f_S(X) 2^{-|S|} = 2^m f(x_1, \ldots, x_N, 1/2, \ldots, 1/2).$$

Since $f(x_1, \ldots, x_N, 1/2, \ldots, 1/2)$ is a projection of $f(X, Y)$, we see that if f is computed by a multilinear circuit C, then C' obtained by setting all the y_i variables to $1/2$ is also multilinear (the polynomials at each node are projections of the respective polynomials in C). Multiplying the output of C' with 2^m gives a circuit for h. This observation was also made in [JKRS09]. Note that the same thing can be done for ABPs or formulas, again with just one extra node, and no increase in width.

Char $= 2$ If the field $\mathbb{F}$ has characteristic 2, then a little more work is needed to compute $h(X)$. We see that for $|S| < m$, the contribution from f_S to h vanishes due to characteristic 2, and we are left with

$$h(X) = \sum_{S \subseteq [m]} f_S(X) 2^{m-|S|} = f_{[m]}(x_1, \ldots, x_N).$$

So we need to compute $f_{[m]}(X)$. The polynomial

$$g(X, y) \triangleq f(x_1, \ldots, x_n, y, y, \ldots, y)$$

may be viewed as a univariate polynomial $g'(y)$ in $G[y]$, where G is the ring $\mathbb{F}[X]$. Then $f_{[m]}(X)$ is just the coefficient of y^m in g'. (Note that $g'(y)$ has degree at most m, since f is multilinear in Y.)

If a circuit C of size s computes f, then setting each $y_i \in Y$ to y gives circuit C', also of size s, computing g'. Now there are two cases to consider.

Infinite fields. This is the easier case, and we give a construction that even preserves width, using the standard interpolation trick. Let $g'(y) = \sum_{i=0}^{m} c_i y^i$ where $c_i \in \mathbb{F}[X]$. Pick $m+1$ distinct values α_j from $\mathbb{F}$ and consider the system of equations $\sum_{i=0}^{m} c_i \alpha_j^i = g'(\alpha_j); \quad 0 \leqslant j \leqslant m$. More succinctly, $V[c_0, \ldots, c_m]^t = [g'(\alpha_o), g'(\alpha_1), \ldots, g'(\alpha_m)]^t$, where V is a Vandermonde matrix and is hence invertible. Hence c_m is a linear combination of the polynomials $g'(\alpha_o), g'(\alpha_1), \ldots, g'(\alpha_m)$ computed by distinct copies of C'. By increasing the depth/length, this linear combination can be computed in a way that increases the width of a circuit only by 1 and of an ABP only by 2.

It follows easily that if f is computed by a multilinear circuit, then the circuit obtained above is also multilinear.

Finite fields. Using the standard procedure of homogenization (see [SY10, Mah14]), we can obtain circuits $C_0, C_1, \ldots, C_m$ for the homogeneous components of g'. The circuit $D(X) = C_m(X, 1)$, is the desired circuit for h. The size is bounded by $s(m+1)^2$, and width by $w(m+1)$.

It remains to see why D is multilinear when C is multilinear. For this, we need to look at the structure of the homogeneous circuit obtained by the homogenization procedure. For every gate u in C, we introduce $m+1$ gates in the homogeneous circuit. We refer to these $m+1$ copies as the major gates. Each major gate computes a homogeneous part of the polynomial computed at u in C. Therefore, a major gate corresponds to a gate u in C, and a degree $i \in \{0, 1, \ldots, m\}$; we refer to this gate as $[u, i]$. Hence the output gate of the circuit $D(X)$ is the gate $[r, m]$, where r is the output gate of C. The edge connections in the homogeneous circuit D are defined inductively. For the input gates we label the copies appropriately. If $u = v + z$, we make the connections based on the rule $[u, i] = [v, i] + [z, i]$ for all i. Otherwise if $u = v \times z$, the connections are based on the rule $[u, i] = \sum_{k=0}^{i} [v, k] \times [z, i-k]$. In this last case, we refer to the intermediate (multiplication) gates used at each major gate to accumulate the homogeneous parts as the minor gates.

Let $p_C(X, Y)$ and $p_D(X)$ be the polynomials computed at u in C and at $[u, i]$ in D respectively. We know that $p_C(X, Y)$ is multilinear in X and Y. Hence in the expression $p_C(X, y, y, \ldots, y) = \sum_{j=0}^{m} p_{C,j}(X) y^j$, each $p_{C,j}(X)$ is multilinear. By construction, $p_D(X) = p_{C,i}(X)$; hence it is multilinear.

Now consider the minor gates. It seems possible that two minor gates compute non-multilinear terms that cancel out when accumulated in the major gate. We need to show that this does not happen, and that the minor gates also compute multilinear polynomials.

Lemma 2.6.2. *Let α, β and γ be three gates in C such that $\alpha = \beta \times \gamma$. If the three gates are multilinear, then the variables appearing in the polynomials computed by β and γ are disjoint.*

Proof. With slight abuse of notation, let α, β, γ also denote the polynomials computed by the respective gates. Suppose there exists a variable v which is in β and in γ. Consider the total order on the variables: $x_1 < \ldots < x_n < y_1 < \ldots < y_m$ and the derived lexicographic order on the monomials. Let m_β and m_γ be the maximal monomials in β and γ which contain v. We show that the monomial $m_\beta m_\gamma$ is in α. If it is not, it is cancelled by some other monomial $n_\beta n_\gamma$. But, $m_\beta m_\gamma$ contains v^2, so n_β and n_γ must both contain v. By assumption, $n_\beta \leqslant m_\beta$ and $n_\gamma \leqslant m_\gamma$. So, $n_\beta n_\gamma \leqslant m_\beta m_\gamma$ and the equality holds only if $m_\beta = n_\beta$ and $m_\gamma = n_\gamma$. Thus the monomial $m_\beta m_\gamma$ does not get cancelled and appears in α. Hence α is not multilinear, a contradiction. □

Lemma 2.6.3. *If $[u, i]$ is a major gate in D, then every variable appearing in the polynomial computed at $[u, i]$ also appears in the polynomial computed at the gate u in C.*

Proof. As before, let $p_D(X)$ and $p_C(X, Y)$ be the polynomials computed at the gates $[u, i]$ and u. Then, $p_D(X)$ is the coefficient of y^i in the polynomial $p_C(x_1, \ldots, x_n, y, \ldots, y)$. Consequently, the variables of $p_D(X)$ are included in the variables of $p_C(x_1, \ldots, x_n, y, \ldots, y)$. Moreover, as $p_C(x_1, \ldots, x_n, y, \ldots, y)$ is a projection of the polynomial $p_C(X, Y)$, the variables of $p_C(x_1, \ldots, x_n, y, \ldots, y)$ are included in the variables in $p_C(X, Y)$. □

A minor gate in D feeding into gate $[u, i]$ corresponds to some $k \in \{0, 1, \ldots, i\}$ and computes $[v, k] \times [z, i - k]$. We have already seen that the polynomials computed at the major gates $[v, k]$ and $[z, i - k]$ are multilinear. Furthermore, by Lemma 2.6.3, the variables of $[v, k]$ (respectively $[z, i - k]$) are a subset of the variables of v (respectively z). As v and z share no variables (Lemma 2.6.2), the same holds for $[v, k]$ and $[z, i - k]$ as well. Thus their product is also multilinear.

This completes the proof for circuits. The same homogenization trick works for ABPs as well, taking size s and width w to size $s(m + 1)$ and width $w(m + 1)$. However for formulas, it could result in a huge blowup in size. But recall that multilinear formulas can be made syntactic multilinear without any increase in size [Raz06]. Hence the result for multilinear formulas follows directly from [JMR13]. □

From Theorem 2.6.1 we obtain the closure property of multilinear classes.

Corollary 2.6.4. *The following circuit classes are closed under exponential sums:*

- *multilinear poly-size bounded-width branching programs (m-*VBWBP*),*
- *multilinear poly-size formulas (m-*VF*),*
- *multilinear poly-size branching programs (m-*VBP*), and*
- *multilinear poly-size circuits (m-*VP*).*

*In particular, we have m-*VP *$=$ m-*VNP.

2.7 Conclusion

In this chapter, we studied reductions in the algebraic setting and proved lower bounds against them. We also studied the closure property of (multilinear) algebraic classes under the exponential summation.

An obvious open question falls from the proof of VNP-hardness of sym-Perm:

- *Can we bring down the number of summands, in Theorem 2.4.6, from 2 to 1?*

This is equivalent to asking whether a linear combination of permanent of *symmetric* matrices can be written as the permanent of a *symmetric* matrix, that is not too large in dimensions. The corresponding statement for the permanent of *arbitrary* matrices follows from VNP-completeness of Perm (with respect to p-projections).

The study of monotone projections raises many interesting questions on monotone projections and Newton polytope (see [Gro15]). In particular, it raises questions about completeness under monotone projections. Formally,

- *Is every non-negative polynomial (i.e. coefficients are non-negative) in* VNP *a monotone projection of the Hamiltonian cycle family* HC_n*? Is there any family of polynomials with such a property?*

Also, the main open question of improving the lower bound for computing the Perm_n over the Boolean $\{\wedge, \vee\}$-semi-ring remains.

Chapter 3

Homomorphism polynomials and Arithmetic classes

3.1 Introduction

One of the most important open questions in algebraic complexity theory is to decide whether the classes VP and VNP are distinct. The significance also comes from the fact that separating them is essential for separating P from NP (at least non-uniformly and assuming the generalised Riemann Hypothesis, over the field $\mathbb{C}$). For details, see Section 4.2 in [Bür00a]. This leading open question of VP versus VNP is often phrased as the permanent versus the determinant problem, since the determinant family is complete for VP. However, as mentioned in Theorem 2.2.11, the hardness of the determinant for VP is only under the more powerful quasi-polynomial-size projections. Under polynomial projections, the determinant is complete for the possibly smaller class VBP. This naturally raises the question of finding polynomials which are complete for VP under polynomial-size projections. Ad hoc families of generic polynomials can be constructed that are VP-complete, but, surprisingly, there are no known natural polynomial families that are VP-complete. Since complete problems characterise complexity classes, the existence of natural complete problems lends added legitimacy to the study of a class. It also shows the robustness (of the definition) of the class by offering an alternative point of view on it that is independent of the choice of a machine model. The determinant and the permanent make the classes VBP, VNP interesting; analogously, what characterises VP?

Unfortunately, very little is known about VP-completeness. The very first polynomial shown to be VP-complete, in [vzG87], was motivated by the definition of VP. (They attributed the result to Fich et al. [FvzGR86].) Indeed the polynomials were so constructed that every polynomial of degree at most n over n variables is a projection of the n-th polynomial in the family. von zur Gathen [vzG87] explicitly stated the question of finding "natural families" that are VP-complete. Then, in [Bür00a], Bürgisser showed that a generic polynomial family constructed recursively while controlling the degree is complete for VP. The construction directly follows a topological sort of a generic VP circuit. In fact, he showed something even more general. He established complete polynomials for each relativised class VP^h, where h is a p-family. (see Section 5.6 in [Bür00a].) In [Raz10] (see also [SY10]), Raz used the depth-reduction of [VSBR83] to show that a family of "universal circuits" is VP-complete; any VP computation can be embedded into it by appropriately setting the variables. All three of these VP-complete families are thus directly obtained using the circuit definition / characterization of VP. In [Men11], Mengel described a way of associating polynomials with constraint satisfaction programs CSPs, and showed that for CSPs where all constraints are binary and the underlying constraint graph is a tree, these polynomials are in VP. Further, for each polynomial in VP, there is such a CSP giving rise to the same polynomial. This means that for the CSP corresponding to the generic VP polynomial or universal circuit, the associated polynomial is VP-complete. The unsatisfactory element here is that to describe the complete polynomial, one again has to fall back to the circuit definition of VP. Similarly, in [CDM13], it is shown that tensor formulas can be computed in VP and can compute all polynomials in VP. Again, to put our hands on a specific VP-complete tensor formula, we need to fall back to the circuit characterisation of VP.

In this chapter, we provide a host of natural families of polynomials that (1) are defined independently of the circuit definition of VP, and (2) are VP-complete. All these families are instances of *homomorphism polynomials*, defined in Definition 3.2.2. We further show that homomorphism polynomials are rich enough to characterise VBP and VNP as well.

We give the basic definitions in Section 3.2. We then discuss upper bounds on their algebraic complexity in Section 3.3. In Section 3.4, Section 3.5, and Section 3.6 we establish hardness for VP, VBP, and VNP respectively.

3.2 Preliminaries

We use (u, v) to denote an undirected edge between u and v, and $\langle u, v\rangle$ to denote a directed edge from u to v. We start with the notion of homomorphisms.

Definition 3.2.1 (Graph Homomorphisms)**.** *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two undirected graphs. A* homomorphism *from G to H is a mapping $\phi : V(G) \to V(H)$ such that the image of an edge is an edge, i.e., for all $(u, v) \in E(G)$, $(\phi(u), \phi(v)) \in E(H)$.*

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

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

The polynomials we consider are defined formally as follows.

Definition 3.2.2. *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two graphs. Consider the set of variables $\overline{Z} := \{Z_{u,a} \mid u \in V(G) \text{ and } a \in V(H)\}$ and $\overline{Y} := \{Y_{(u,v)} \mid (u, v) \in E(H)\}$. Let $\mathcal{H}$ be a set of homomorphisms from G to H. The* homomorphism polynomial *$f_{G,H,\mathcal{H}}$ in the variable set $\overline{Y}$, and the* generalised *homomorphism polynomial $\hat{f}_{G,H,\mathcal{H}}$ in the variable* *set $\overline{Z} \cup \overline{Y}$, are defined as follows:*

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

$$\hat{f}_{G,H,\mathcal{H}} = \sum_{\phi \in \mathcal{H}} \left(\prod_{u \in V(G)} Z_{u,\phi(u)} \right) \left(\prod_{(u,v)\in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

Let **Hom** *denote the set of all homomorphisms from G to H. If $\mathcal{H}$ equals* **Hom***, then we drop it from the subscript and write $f_{G,H}$ or $\hat{f}_{G,H}$.*

We take a moment to emphasise that when we consider directed graphs, $\overline{Y}$ is the set of variables associated with directed edges, and the product in the definition of the polynomials runs over all directed edges of G.

To obtain families of polynomials from the homomorphism polynomial we consider two sequences of graphs (G_m) and (H_m). Then the families are defined to be either $(f_{G_m,H_m,\mathcal{H}})$, or $(\hat{f}_{G_m,H_m,\mathcal{H}})$. A sequence (G_m) of graphs is called a p-family if the number of vertices in G_m is p-bounded in m.

Remark 3.2.1. *For every $G, H, \mathcal{H}$, $f_{G,H,\mathcal{H}}(\overline{Y})$ equals $\hat{f}_{G,H,\mathcal{H}}(\overline{Z}, \overline{Y}) \mid_{\overline{Z}=\overline{1}}$. Thus upper bounds for $\hat{f}$ give upper bounds for f, while lower bounds for f give lower bounds for $\hat{f}$.*

We consider the pathwidth and treewidth parameters for a graph. We will work with a "canonical" form of decompositions which is generally useful in dynamic-programming algorithms. For a detailed treatment of dynamic-programming on tree (path) decomposition see [CFK+15].

Definition 3.2.3. *A* (nice) path decomposition *of a graph G is a sequence of bags $\mathcal{P} = \langle B_1, B_2, \ldots, B_\ell \rangle$, where for all $i \in [\ell]$ $B_i \subseteq V(G)$, such that the following conditions hold:*

1. *$|B_1| = 1$, $B_\ell = \emptyset$, and $\cup_{i\in[\ell]} B_i = V(G)$. That is, every vertex of G is contained in at least one bag.*

2. *For every $(u, v) \in E(G)$, there exists $i \in [\ell]$ such that $\{u, v\} \subseteq B_i$.*

3. *For every $u \in V(G)$, if $u \in B_i \cap B_k$, then $u \in B_j$ for all $i \leqslant j \leqslant k$.*

4. *For every $i \in [2, \ell]$, B_i is one of the following types:*

   - ***Introduce node:*** *$B_i = B_{i-1} \cup \{v\}$ for some vertex $v \notin B_{i-1}$. We say that v is* introduced *at i.*

   - ***Forget node:*** *$B_i = B_{i-1} \setminus \{w\}$ for some vertex $w \in B_{i-1}$. We say that w is* forgotten *at i.*

The *width* of a path decomposition $\mathcal{P}$ is one less than the size of the largest bag; that is, $\max_{i \in [\ell]} |B_i| - 1$. The *path-width* of a graph G is the minimum possible width of a path decomposition of G, and is denoted $pw(G)$. From the definition of nice path decomposition, it follows that every vertex of G gets introduced and becomes forgotten exactly once, hence the total number of bags in the sequence $\mathcal{P}$ is exactly $2|V(G)|$. We now define tree decomposition which is a generalisation of a path decomposition.

Definition 3.2.4. *A* (nice) tree decomposition *of a graph G is a pair $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$, where T is a tree, rooted at B_r, whose every node t is assigned a set $B_t \subseteq V(G)$, such that the following conditions hold:*

1. *$B_r = \emptyset$, $|B_\ell| = 1$ for every leaf ℓ of T, and $\cup_{t \in V(T)} B_t = V(G)$.*
   That is, the root contain the empty bag, the leaves contain singleton sets, and every vertex of G is in at least one bag.

2. *For every $(u, v) \in E(G)$, there exists a node t of T such that $\{u, v\} \subseteq B_t$.*

3. *For every $u \in V(G)$, the set $T_u = \{t \in V(T) \mid u \in B_t\}$ induces a connected subtree of T.*

4. *Every non-leaf node t of T is of one of the following three types:*

   - ***Introduce node:*** *t has exactly once child t', and $B_t = B_{t'} \cup \{v\}$ for some vertex $v \notin B_{t'}$. We say that v is* introduced *at t.*

- ***Forget node:*** *t has exactly one child t', and $B_t = B_{t'} \setminus \{w\}$ for some vertex $w \in B_{t'}$. We say that w is* forgotten *at t.*

- ***Join node:*** *t has two children t_1, t_2, and $B_t = B_{t_1} = B_{t_2}$.*

The *width* of a tree decomposition $\mathcal{T}$ is one less than the size of the largest bag; that is, $\max_{t \in V(T)} |B_t| - 1$. The *tree-width* of a graph G, denoted $tw(G)$, is the minimum possible width of a tree decomposition of G. It can be shown (see Lemma 7.4 in [CFK+15]) that for any G, there exists a nice tree decomposition that has at most $O(tw(G)|V(G)|)$ nodes in the tree. Furthermore, observe that a path decomposition can be thought of as a tree decomposition with *no* join nodes. It is known that $pw(G) = O(\log n \cdot tw(G))$ [KS93].

A sequence (G_m) of graphs is said to have *bounded* tree(path)-width if for some absolute constant c independent of m, the tree(path)-width of each graph in the sequence is bounded by c.

We now mention some structural properties of arithmetic circuits that would be useful in establishing hardness of certain polynomials. In the context of arithmetic circuits, the notion of *parse trees* is used to certify that a particular monomial is generated during the computation. Parse trees have been studied under different names [AJMV98, JS82, VT89, Ven92, MVW04, MP08]. We will work with the following definition from [MP08].

Definition 3.2.5 (Parse trees)**.** *The set of parse trees of a circuit C is defined inductively:*

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

- *If the output gate of C is a $\times$ gate whose children are the gates α and β, the parse trees of C are obtained by taking a parse tree of the subtree rooted at α, a parse tree of a disjoint copy of the subtree rooted at β and the edges from α and β to the output gate.*

- *If the output of C is a $+$ gate, the parse trees of C are obtained by taking a parse tree of a subcircuit rooted at one of the children and the edge from the (chosen)* *child to the output gate.*

Each parse tree T is associated with a monomial $\mathsf{mon}(T)$, which is obtained by computing the product of the labels of the input gates that appear in T. The following lemma establishes a formal connection between the polynomial computed and the parse trees of the circuit. A proof is easily seen via induction.

Lemma 3.2.6 ([MP08]). *Let $f(\bar{x})$ be a polynomial computed by a circuit C. Then, $f(\bar{x}) = \sum_T \mathsf{mon}(T)$, where the sum is over the set of parse trees T of C.*

A circuit is said to be *multiplicatively disjoint* circuit if for any multiplication gate in the circuit, the subcircuits rooted at its children are disjoint, i.e., they do not share any vertex. The next proposition states a particularly useful property of multiplicative disjoint circuits.

Proposition 3.2.7 ([MP08]). *A circuit C is multiplicatively disjoint if and only if any parse tree of C is a subgraph of C. Furthermore, a subgraph T of C is a parse tree if the following conditions are met:*

- *T contains the output gate of C.*
- *If α is a multiplication gate in T having gates β and γ as children in C, then the edges $\langle \beta, \alpha \rangle$ and $\langle \gamma, \alpha \rangle$ also appear in T.*
- *If α is an addition gate in T, it has only one child in T.*
- *Only edges and gates obtained in this way belong to T.*

Moreover, it is known that if we are dealing with arithmetic circuits such that their size and degree are polynomially bounded, then, without loss of generality, we can assume the circuit to be depth reduced and multiplicatively disjoint.

Proposition 3.2.8 ([VSBR83, MP08]). *If (f_n) is in VP, then f_n can be computed by a polynomial-size circuit of depth $O(\log n)$ where $+$ gates are allowed to have unbounded* *fan-in, but each $\times$ gate has fan-in at most 2. Furthermore, the circuit is multiplicatively disjoint.*

Raz [Raz10] studied a family (D_n) of **universal circuits** computing a polynomial family (p_n), see also [SY10]. These circuits are universal in the sense that every polynomial $f_n(x_1, \ldots, x_n)$ of degree d, computed by a circuit of size s, can be computed by a circuit Ψ such that the underlying graph of Ψ is the same as the graph of D_m, for $m \in \mathsf{poly}(n, s, d)$. (In fact, f_n can be obtained as a projection of p_m.) With minor modifications to (D_n) (simple padding with dummy gates, followed by the multiplicative disjointness transformation from [MP08]), we can show that there is a universal circuit family $(\mathfrak{U}_n)$ in the normal form described below:

Definition 3.2.9 (Normal Form Universal Circuits)**.** *A universal circuit* $(\mathfrak{U}_n)$ *in normal form is a circuit with the following structure:*

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

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

The following well-known technical lemma allows us to use interpolation to extract the coefficient of a particular monomial of a polynomial. We say that a multivariate polynomial $f(\bar{x})$ has degree d in x_i iff x_i has degree at most d in every monomial of $f(\bar{x})$.

Lemma 3.2.10 (folklore)**.** *Suppose* $\mathbb{F}$ *is a field with characteristic* 0*. Let* $f(\bar{x}, y_1, \ldots, y_\ell)$ *be a polynomial in* $\mathbb{F}[\bar{x}, y_1, \ldots, y_\ell]$*. Further, assume* f *has degree* D_i *in* y_i*, for* $i \in [\ell]$*. Let* $\sum_{\tilde{d}=(d_1,\ldots,d_\ell)} f_{\tilde{d}}(\bar{x}) \prod_{i=1}^{\ell} y_i^{d_i}$ *be the representation of* f *when viewed as a polynomial in* $\mathbb{F}[\bar{x}][y_1, \ldots, y_\ell]$*. Then, for any* $\tilde{d}$*,* $f_{\tilde{d}}(\bar{x})$ *can be written as a linear combination of* $\prod_{i=1}^{\ell}(D_i + 1)$ *many projections of* f*.*

Proof. The proof easily follows by induction on ℓ. We only sketch the proof here. The essence of the proof is in the base case $\ell = 1$, which we now illustrate.

Fix $\ell = 1$. Let the degree of y in $f(\bar{x}, y)$ be D, that is, $f(\bar{x}, y) = \sum_{i=0}^{D} f_i(\bar{x}) y^i$. Let $\alpha_0, \ldots, \alpha_D$ be any $D + 1$ non-zero distinct points in $\mathbb{F}$. We consider the following system of linear equations in the coefficient $f_i(\bar{x})$:

$$\begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^D \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_D & \alpha_D^2 & \cdots & \alpha_D^D \end{bmatrix} \begin{bmatrix} f_0(\bar{x}) \\ f_1(\bar{x}) \\ \vdots \\ f_D(\bar{x}) \end{bmatrix} = \begin{bmatrix} f(\bar{x}, \alpha_0) \\ f(\bar{x}, \alpha_1) \\ \vdots \\ f(\bar{x}, \alpha_D) \end{bmatrix}.$$

The $(D + 1) \times (D + 1)$ matrix on the left side of the aforementioned system of linear equations is called a *Vandermonde matrix*. We denote it by $V(\bar{\alpha})$. It is known that $\mathsf{det}(V(\bar{\alpha})) = \prod_{0 \leqslant i < j \leqslant D}(\alpha_i - \alpha_j)$, and since α_i's are distinct, this determinant is non-zero.

Thus, the above system of linear equations has a unique solution given by,

$$\begin{bmatrix} f_0(\bar{x}) \\ f_1(\bar{x}) \\ \vdots \\ f_D(\bar{x}) \end{bmatrix} = V(\bar{\alpha})^{-1} \begin{bmatrix} f(\bar{x}, \alpha_0) \\ f(\bar{x}, \alpha_1) \\ \vdots \\ f(\bar{x}, \alpha_D) \end{bmatrix}.$$

Therefore, it follows that each $f_i(\bar{x})$ can be written as a linear combination of $D + 1$ projections of $f(\bar{x}, y)$, where each projection is $f(\bar{x}, y)|_{y=D_i}$ for $0 \leqslant i \leqslant D$. □

3.3 Upper Bounds

In this section, we show that for any p-family (H_n), and any bounded tree-width (path-width, respectively) p-family (G_n), the polynomial family (f_n) where $f_n = \hat{f}_{G_n,H_n}$ is in VP (VBP, respectively). Following Remark 3.2.1 it suffices to consider upper bounds for the generalised homomorphism polynomial $\hat{f}_{G,H}$. Moreover, for the ease of presentation we will consider $\mathcal{H} = \textbf{Hom}$. If we want to consider a restricted set $\mathcal{H}$ of homomorphisms, such as directed homomorphisms, all we need is that homomorphisms in $\mathcal{H}$ can be obtained from independent parts with a local stitching-together operator. That is, $\phi \in \mathcal{H}$ can be verified locally edge-by-edge and/or vertex-by-vertex, so that this can be built into the inductive construction.

We start with an easy observation that says homomorphism polynomials are *explicit*, that is they belong to the class VNP.

Proposition 3.3.1. *Let (G_n) and (H_n) be p-families of graphs. Consider the family of homomorphism polynomial (f_n), where $f_n = \hat{f}_{G_n,H_n}(\overline{Z}, \overline{Y})$. Then, $(f_n) \in$ VNP.*

Proof. It follows straightforwardly from Valiant's criterion, Proposition 2.2.12. □

We now state and prove the main algorithm of this section.

Lemma 3.3.2. *Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two graphs. Then the generalised homomorphism polynomial $\hat{f}_{G,H}$ is computable by an arithmetic circuit of size $O\Big(tw(G) \cdot |V(G)| \cdot |V(H)|^{tw(G)+1}(|V(H)| + |E(H)|)\Big)$, where $tw(G)$ is the tree-width of G.*

Proof. Let $\mathcal{T} = (T, \{B_t\}_{t \in V(T)})$ be a nice tree decomposition of G of width τ. For each $t \in V(T)$, let $M_t = \{\phi \mid \phi \colon B_t \to V(H)\}$ be the set of all mappings from B_t to $V(H)$. Since $|B_t| \leqslant \tau + 1$, we have $|M_t| \leqslant |V(H)|^{\tau+1}$. For each node $t \in V(T)$, let T_t be the subtree of T rooted at node t, $V_t := \bigcup_{t' \in V(T_t)} B_{t'}$, and $G_t := G[V_t]$ be the subgraph of G induced on V_t. Note that $G_r = G$.

We will build the circuit inductively. For each $t \in V(T)$ and $\phi \in M_t$, we have a gate $\langle t, \phi \rangle$ in the circuit. Such a gate will compute the homomorphism polynomial $f_{G_t,H,\mathcal{H}}$ from G_t to H such that $\mathcal{H}$ is the set of those homomorphisms which agree with ϕ on B_t. For each such gate $\langle t, \phi \rangle$ we introduce another gate $\langle t, \phi \rangle'$ which computes the "partial derivative" (or, quotient) of the polynomial computed at $\langle t, \phi \rangle$ with respect to the monomial given by ϕ. As we mentioned before, the construction is inductive, starting at the leaf nodes and proceeding towards the root.

Base case (Leaf nodes): Let $\ell \in V(T)$ be a leaf node. Then, $B_\ell = \{u\}$ for some $u \in V(G)$. Note that any $\phi \in M_\ell$ is just a mapping of u to some node in $V(H)$. Hence, the set M_ℓ can be identified with $V(H)$. Therefore, for all $h \in V(H)$, we label the gate $\langle \ell, h \rangle$ by the variable $Z_{u,h}$. The derivative gate $\langle \ell, h \rangle'$ in this case is set to 1.

Introduce nodes: Let $t \in V(T)$ be an introduce node, and t' be its unique child. Then, $B_t \setminus B_{t'} = \{u\}$ for some $u \in V(G)$. Let $N(u) := \{v | v \in B_{t'} \text{ and } (v, u) \in E(G_t)\}$. Note that there is a one-to-one correspondence between $\phi \in M_t$ and pairs $(\phi', h) \in M_{t'} \times V(H)$.

Therefore, for all $\phi(= (\phi', h)) \in M_t$ if $\forall v \in N(u), (\phi'(v), h) \in E(H)$, then we set

$$\langle t, \phi \rangle := Z_{u,h} \cdot \left(\prod_{v \in N(u)} Y_{(\phi'(v),h)} \right) \cdot \langle t', \phi' \rangle \qquad \text{and,}$$
$$\langle t, \phi \rangle' := \langle t', \phi' \rangle',$$

otherwise we set $\langle t, \phi \rangle = \langle t, \phi \rangle' := 0$.

Forget nodes: Let $t \in V(T)$ be a forget node and t' be its unique child. Then, $B_{t'} \setminus B_t = \{u\}$ for some $u \in V(G)$. Again note that there is a one-to-one correspondence between pairs $(\phi, h) \in M_t \times V(H)$ and $\phi' \in M_{t'}$. Let $N(u) := \{v | v \in B_t \text{ and } (v, u) \in E(G_{t'})\}$. Therefore, for all $\phi \in M_t$, we set

$$\langle t, \phi \rangle := \sum_{h \in V(H)} \langle t', (\phi, h) \rangle \qquad \text{and,}$$
$$\langle t, \phi \rangle' := \sum_{\substack{h \in V(H) \text{ such that} \\ \forall v \in N(u),\ (\phi(v), h) \in E(H)}} Z_{u,h} \cdot \left(\prod_{v \in N(u)} Y_{(\phi(v),h)} \right) \cdot \langle t', (\phi, h) \rangle'.$$

Join nodes: Let $t \in V(T)$ be a join node, and t_1 and t_2 be its two children; we have $B_t = B_{t_1} = B_{t_2}$. Then, for all $\phi \in M_t$, we set

$$\langle t, \phi \rangle := \langle t_1, \phi \rangle \cdot \langle t_2, \phi \rangle' \; (= \langle t_1, \phi \rangle' \cdot \langle t_2, \phi \rangle)$$
$$\langle t, \phi \rangle' := \langle t_1, \phi \rangle' \cdot \langle t_2, \phi \rangle'.$$

The output gate of the circuit is $\langle r, \emptyset \rangle$. The correctness of the algorithm is readily seen via induction in a similar way. The bound on the size follows, since $|V(T)| = O(tw(G)|V(G)|)$, $|M_t| \leqslant |V(H)|^{\tau+1}$, and implementing each node may need $O(|V(H)| + |E(H)|)$ extra gates. □

Remark 3.3.1. *We note that the circuit constructed is a constant-free circuit, i.e., it only use constants from the set* $\{0, 1\}$. *Further, if we start with a path decomposition, we obtain* *a* skew *circuit, since the* join *nodes are absent.*

From Lemma 3.3.2 and the remark above, we obtain the following theorem which improves upon the obvious bound of Proposition 3.3.1, when tree decompositions of G are of special kind.

Theorem 3.3.3. *Consider the family of homomorphism polynomials* (f_n)*, where* $f_n = \hat{f}_{G_n,H_n}(\overline{Z}, \overline{Y})$*, and* (H_n) *is a p-family of complete graphs.*

- *If* (G_n) *is a p-family of graphs of bounded tree-width, then* $(f_n) \in \mathsf{VP}$.
- *If* (G_n) *is a p-family of graphs of bounded path-width, then* $(f_n) \in \mathsf{VBP}$.

3.4 Completeness : VP

In this section we will characterise the algebraic class VP using homomorphism polynomials. In particular, we will establish that there exists a p-family (G_n) of graphs of bounded tree-width such that the polynomial f_{G_n,K_m} (cf. Definition 3.2.2), for $m \in \mathsf{poly}(n)$, is complete for VP with respect to p-projections. The membership in VP follows directly from Theorem 3.3.3. Thus, henceforth, our main objective is to establish *hardness*.

Let us consider the universal circuit $\mathfrak{U}_n$ in normal form (Definition 3.2.9). From inspection it follows that the parse trees of $\mathfrak{U}_n$ are isomorphic to the following graph: a directed balanced alternately-binary-unary tree with depth $2k(n)$. Vertices on an odd layer have exactly two incoming edges whereas vertices on an even layer have exactly one incoming edge. The first layer has only one vertex called root, and the edges are directed from leaves towards the root. Furthermore, because of multiplicative disjointness, we know parse trees are subgraphs of $\mathfrak{U}_n$.

Hence, the observation suggests a way to capture monomial computations of the universal circuit via homomorphisms from the directed balanced alternately-binary-unary tree into

$\mathfrak{U}_n$. In fact, we will go a step further and consider homomorphisms from undirected complete binary trees.

3.4.1 Homomorphism with weights

For m a power of 2, let T_m denote a complete (perfect) binary tree with m leaves. We recall the depth of $\mathfrak{U}_n = 2k(n) = 2c\lceil \log n \rceil$, for some $c > 0$, and $\mathsf{size}(\mathfrak{U}_n) = s_n$ where s_n is p-bounded in n.

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

More precisely, we obtain a sequence of graphs (J_n) from the undirected graphs underlying $(\mathfrak{U}_n)$. To make the presentation clearer, we first construct an intermediate graph J'_n as follows. Retain the multiplication and input gates of $\mathfrak{U}_n$. Let us make two copies of each. For each retained gate, g, in $\mathfrak{U}_n$; let g_L and g_R be the two copies of g in J'_n (see Figure 3.1). The two copies, g_L and g_R, will be used to connect to a grandparent from left and right, respectively. We now define the edge connections in J'_n. Assume g is a $\times$ gate retained in J'_n. Let α and β be two + gates feeding into g in $\mathfrak{U}_n$. Let $\{\alpha_1, \ldots, \alpha_i\}$ and $\{\beta_1, \ldots, \beta_j\}$ be the gates feeding into α and β, respectively. Assume without loss of generality that α and β feed into g from left and right, respectively. Now we add the following sets of edges to

J'_n:

$$\{(\alpha_{1L}, g_L), \ldots, (\alpha_{iL}, g_L)\} \cup \{(\beta_{1R}, g_L), \ldots, (\beta_{jR}, g_L)\},$$
$$\text{and } \{(\alpha_{1L}, g_R), \ldots, (\alpha_{iL}, g_R)\} \cup \{(\beta_{1R}, g_R), \ldots, (\beta_{jR}, g_R)\}.$$

We now would like to keep a single copy of $\mathfrak{U}_n$ in these sets of edges. So we remove the vertex $root_R$ and we remove the remaining spurious edges in following way. If we assume that all edges are directed from root towards leaves, then we keep only edges induced by the vertices reachable from $root_L$ in this directed graph.

We now transform J'_n as follows to get J_n (cf. Figure 3.1): for each gate g' in J'_n which corresponds to an input gate in $\mathfrak{U}_n$, we add two new distinct vertices and connect them to g'. Note that there are two type of vertices in J_n; one that corresponds to a gate in $\mathfrak{U}_n$ and others are degree 1 vertices hanging from gates that correspond to input gates in $\mathfrak{U}_n$.

Observation 3.4.1. *There is a one-to-one correspondence between parse trees of $\mathfrak{U}_n$ and subgraph of J_n that are rooted at $root_L$ and isomorphic to $\mathsf{T}_{2^{k(n)+1}}$.*

Based on the observation we would like to capture the parse trees of $\mathfrak{U}_n$ via homomorphisms from $\mathsf{T}_{2^{k(n)+1}}$ into $\mathfrak{U}_n$. But we need to be careful because there are far more homomorphisms than parse trees. So we consider the following weighted variant of the homomorphism polynomials.

Let $G = (V(G), E(G))$ and $H = (V(H), E(H))$ be two graphs. Let $\alpha : V(G) \to \mathbb{N}$ be a labeling of vertices of G by non-negative integers. Consider the set of variables $\overline{X} := \{X_u \mid u \in V(H)\}$ and $\overline{Y} := \{Y_{(u,v)} \mid (u,v) \in E(H)\}$. The *weighted* homomorphism polynomial $f^{\alpha}_{G,H}$ in the variable set $\overline{X} \cup \overline{Y}$ is defined as follows:

$$f^{\alpha}_{G,H} = \sum_{\phi \in \mathbf{Hom}} \left(\prod_{u \in V(G)} X^{\alpha(u)}_{\phi(u)} \right) \left(\prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

However, for our purposes, $\{0, 1\}$-valued weights suffices, i.e., $\alpha : V(G) \to \{0, 1\}$. Such

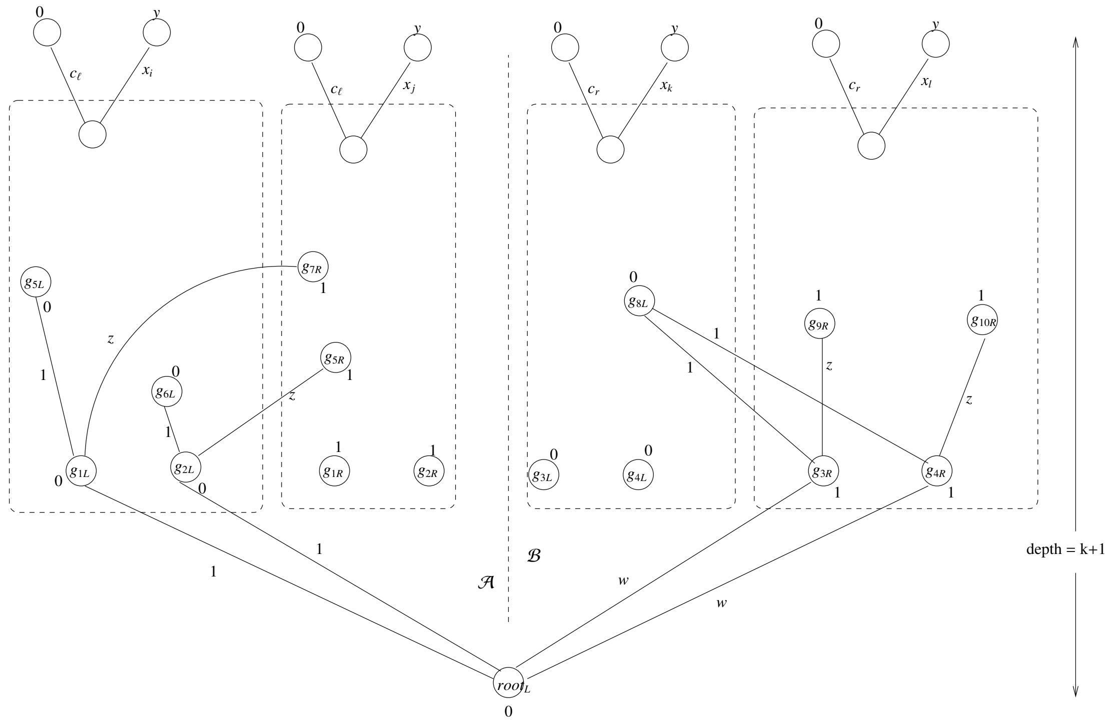

Figure 3.1: Graph J_n with vertex and edge labels

$\{0,1\}$-valued weights are commonly used in the literature, see, e.g., [BCL+06]. Thus, in our setting, $f_{G,H}$ is a projection of $f^{\alpha}_{G,H}$ (set all variables in $\overline{X}$ to 1) which, in turn, is a projection of $\hat{f}_{G,H}$ (set $Z_{u,a}$ to X_a if $\alpha(u) = 1$, and 1 if $\alpha(u) = 0$). Indeed, the hardness for $f_{G,H}$, which we promised in the introduction (and will show later), establishes the hardness for $f^{\alpha}_{G,H}$. But our purpose here is to draw the motivation for the harder case of $f_{G,H}$. For $\{0,1\}$-valued weights α, $f^{\alpha}_{G,H}$ equals

$$\sum_{\phi \in \mathbf{Hom}} \left(\prod_{\substack{u \in V(G) \\ \alpha(u)=1}} X_{\phi(u)} \right) \left(\prod_{(u,v) \in E(G)} Y_{(\phi(u),\phi(v))} \right).$$

We now state and prove the main theorem of this subsection.

Theorem 3.4.1. *Over fields of characteristic 0, the family of homomorphism polynomials (f_m), with $f_m(\overline{X}, \overline{Y}) = f^{\alpha}_{G_m,H_m}(\overline{X}, \overline{Y})$, where*

- $G_m := \mathsf{T}_m$.

- *H_m is an undirected complete graph on* $\mathsf{poly}(m)$*, say* m^6*, nodes.*

- *Define* $\alpha : \mathsf{T}_m \to \{0, 1\}$ *such that,*

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

is complete for VP *with respect to linear p-projections.*

Since the proof is long with several case analysis, we would like to discuss the proof outline before presenting the proof.

We use $\overline{Y}$ variables to pick out J_n from H_m. We assign special variables w on edges from the root to a node g_R, and z on edges going from a non-root non-input node u to some right copy node g_R (see Fig. 3.1). For an input node g in the "left sub-graph" of J_n, the new left and right edges are assigned c_ℓ and x respectively, where x is the corresponding input label of g in $\mathfrak{U}_n$, and the node at the end of the x edge is assigned a special variable y. Similarly, in the right sub-graph variables c_r, x and y are used with c_r replacing c_ℓ.

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

Proof. As mentioned before, the membership in VP follows from Theorem 3.3.3 and the fact that $f^{\alpha}_{G,H}$ is a projection of $\hat{f}_{G,H}$. We now present the hardness proof. Before starting the proof, we set up the notation.

Let us set $m := 2^{k(n)+1}$. The choice of $\mathsf{poly}(m)$ is such that $4s_n \leqslant \mathsf{poly}(m)$, where s_n is the size of $\mathfrak{U}_n$. (In particular, there exist universal circuits such that $s_n = O(n^6)$. Hence, the choice of $\mathsf{poly}(m)$ could be roughly m^6.) The $\overline{Y}$ variables are set to $\{0, 1, w, z, c_\ell, c_r, \bar{x}\}$ such that the edges set to non-zero together form the graph J_n. The $\overline{X}$ variables take values in $\{0, 1, y\}$. The variables on nodes corresponding to the left copies of gates in $\mathfrak{U}_n$ are set to 0, whereas those on the right copies are set to 1. The $\overline{X}$ variables for degree 1 vertices hanging from input gates are set to 0 or 'y' depending on whether they are left or right child, respectively.

For every edge $(root_L, g_R)$, we set $Y_{(root_L, g_R)} := w$. For all $u \in V(J_n)$, except $root_L$, with degree of u not equal to 1, if the edge (u, g_R) exist then we set $Y_{(u,g_R)} := z$.

Let v be a gate, in J_n, corresponding to an input gate g in $\mathfrak{U}_n$ and v lies in $\mathcal{A}$ part (see Figure 3.1). Let v_1 and v_2 be the left and right leaf attached to v, then we set $Y_{vv_1} := c_\ell$ and $Y_{vv_2} :=$ the $\bar{x}$-label of g in $\mathfrak{U}_n$.

For v a gate, in J_n, corresponding to an input gate g in $\mathfrak{U}_n$ and lying in $\mathcal{B}$ part (see Figure 3.1), let v_1 and v_2 be the left and right leaf attached to v. Then we set $Y_{vv_1} := c_r$ and $Y_{vv_2} :=$ the $\bar{x}$-label of g in $\mathfrak{U}_n$.

All other remaining edge variables that are not set to 0, are set to 1.

Recall for a parse tree T, by $\mathsf{mon}(T)$ we mean the monomial associated with T. Similarly, for a homomorphism ϕ, $\mathsf{mon}(\phi)$ denotes the monomial $\left(\prod_{u\in V(G)} X_{\phi(u)}^{\alpha(u)}\right)\left(\prod_{(u,v)\in E(G)} Y_{(\phi(u),\phi(v))}\right)$.

By Observation 3.4.1 we easily deduce that for each parse tree T of $\mathfrak{U}_n$ there exist a homomorphism ϕ from $\mathsf{T}_{2^{k(n)+1}}$ to J_n such that $\mathsf{mon}(\phi)$ is equal to $\mathsf{mon}(T)\times\left(wz^{(2^k-2)}c_\ell^{2^{(k-1)}}c_r^{2^{(k-1)}}y^{2^k}\right)$, where $k = k(n)$.

We claim that for a homomorphism ϕ, if $\mathsf{mon}(\phi)$ has degree 1 in w, $(2^k - 2)$ in z, 2^{k-1} in c_ℓ, 2^{k-1} in c_r and 2^k in y, then the homomorphic image $\phi(\mathsf{T}_{2^{k(n)+1}})$ is isomorphic to $\mathsf{T}_{2^{k(n)+1}}$ rooted at $root_L$.

We will prove the claim in two parts. First we prove that if any node other than the root of $\mathsf{T}_{2^{k(n)+1}}$ is mapped to $root_L$ then the corresponding monomial does not have right degree in w, c_ℓ or c_r. We then consider the case where the root of the complete binary tree is the only node mapped to $root_L$ under ϕ, and we argue that if ϕ has the required degrees then it must be a complete binary tree with $2^{k(n)+1}$ leaves rooted at $root_L$.

Case 1: $\phi^{-1}(root_L) = \emptyset$. Clearly $\mathsf{mon}(\phi)$ has degree *zero* in w.

Case 2: $\phi^{-1}(root_L)$ contains a degree 3 vertex, say v. Let v_1 and v_2 be the left and right child of v, respectively. Let v_3 be the parent of v in T_m. Note that v must be labeled 0 for the monomial to survive, since X_{root_L} has been set to 0. Also, at least one of v_1, v_2, and v_3 is labeled 1.

Case 2a: Suppose two of the v_i's are labeled 1. Hence for the $\mathsf{mon}(\phi)$ to survive these v_i's must be mapped to the right of $root_L$. But then $\mathsf{mon}(\phi)$ has degree at least 2 in w.

Case 2b: Exactly one of the v_i is labeled 1. It must be the right child v_2, for the monomial to survive it should be mapped to the right of $root_L$. Now if v_1 or v_3 is also mapped to the right of $root_L$, $\mathsf{mon}(\phi)$ will have degree at least 2 in w. Otherwise, both v_1 and v_3 are mapped to the left of $root_L$. Since v_1 is an internal vertex of T_m, the subtree rooted at v_2 and v_1 has depth at most $k - 1$ in T_m. In the first case $\mathsf{mon}(\phi)$ does not have sufficient degree in c_ℓ , whereas in the second case it does not have sufficient degree in c_r.

Case 3: $\phi^{-1}(root_L)$ contains the root of T_m and at least one degree 1 vertex, say v. Also, no degree 3 vertices are mapped to $root_L$. As before, the left child of the root of T_m is mapped to the left of $root_L$ and the right child is mapped to the right of $root_L$, else either the monomial evaluates to zero or has degree at least 2 in w.

Case 3a: For some leaf node v mapped to $root_L$, its neighbour is mapped to the right of $root_L$. In this case if the monomial is not zero, we will have at least degree 2 in w.

Case 3b: For all leaf node v mapped to $root_L$, their neighbour is mapped to the left of $root_L$. But now $\mathsf{mon}(\phi)$ will not have sufficient degree in c_ℓ.

Case 4: $\phi^{-1}(root_L)$ contains only degree 1 vertices. But then the homomorphic image is confined only to the left side or right side of $root_L$. Hence the monomial will not have sufficient degree in either c_r or c_ℓ.

Therefore, we have shown that to get the appropriate degrees as claimed, $\phi^{-1}(root_L)$ must contain the root of T_m and nothing else. Now to complete the proof we will show that if $\mathsf{mon}(\phi)$ has correct degrees in w, z, c_ℓ, c_r and y, then ϕ is injective and preserves left-right labelling of nodes of T_m. Note that for the monomial to survive and have degree 1 in w, it must be the case that the right child of the root of T_m is mapped to the right of $root_L$ and the left child is mapped to the left of $root_L$.

We claim that the homomorphism ϕ can not 'fold back' layers, that is, map a descendant to the node where its ancestor is mapped. This is because otherwise the monomial will not have sufficient degree in either c_ℓ, c_r, or y (if folding happens at depth k+1).

We also claim that the homomorphism ϕ can not 'squish' a layer, that is, map two siblings to the same node. If the two are mapped to a vertex labeled 0, the monomial evaluates to zero. In the other case, they are mapped to a vertex labeled 1 but then the two siblings together, either contribute degree 2 in z or miss out at least degree 1 in c's which cannot be compensated later if the monomial is non-zero.

Therefore we have shown that homomorphisms that are injective, whose image is isomorphic to T_m and rooted at $root_L$, and which preserve left-right labels are in one-to-one correspondence with parse trees of $\mathfrak{U}_n$.

Thus to compute the universal polynomial $\mathsf{f}_{\mathfrak{U}_n}(\bar{x})$ we interpolate (Lemma 3.2.10) the oracle polynomial $f^{\alpha}_{\mathsf{T}_m,H_m}(\bar{x}, w, z, c_\ell, c_r, y)$ to extract the coefficient of $wz^{(2^k-2)}c_\ell^{2^{(k-1)}}c_r^{2^{(k-1)}}y^{2^k}$. Since we are interpolating over 5 variables and the degree in each is polynomially bounded, Lemma 3.2.10 implies that the universal polynomial is a linear combination of polynomially many projections of the weighted homomorphism polynomial.

$\square$

We now proceed to improve upon the above theorem by establishing hardness under p-projections while removing both the restrictions: weights α and variables on vertices $\overline{X}$, i.e., hardness of $f_{G,H}$ rather than $f^{\alpha}_{G,H}$. The price we pay for such a neat form is our source graph G_m gets slightly more non-trivial compared to the simple T_m in Theorem 3.4.1.

3.4.2 The unweighted homomorphism polynomial

In this subsection, we establish the VP-*hardness* of the homomorphism polynomials. We need to show that there exists a p-family (G_m) of bounded tree-width graphs such that $(f_{G_m,H_m}(\overline{Y}))$ is hard for VP under p-projections.

We use *rigid* and mutually *incomparable* graphs in the construction of G_m. A graph is called *rigid* if it has *no* homomorphism to itself other than the identity map. Two graphs G and H are called *incomparable* if there are *no* homomorphisms from $G \to H$ as well as $H \to G$. It is known that asymptotically almost all graphs are rigid, and almost all pairs of nonisomorphic graphs are also incomparable. For the purposes of this paper, we only need a collection of three rigid and mutually incomparable graphs. For more details, we refer to [HN04].

Let $I := \{I_0, I_1, I_2\}$ be a fixed set of three connected, rigid and mutually incomparable graphs. (Later we will describe an explicit set of rigid and mutually incomparable graphs.) Note that they are necessarily *non-bipartite*. Let $c_{I_i} = |V(I_i)|$. Choose an integer $c_{\max} > \max\{c_{I_0}, c_{I_1}, c_{I_2}\}$. Identify two distinct vertices $\{v^0_\ell, v^0_r\}$ in I_0, three distinct vertices $\{v^1_\ell, v^1_r, v^1_p\}$ in I_1, and three distinct vertices $\{v^2_\ell, v^2_r, v^2_p\}$ in I_2.

Recall for every m, a power of 2, T_m denotes a complete (perfect) binary tree with m leaves. We construct a sequence of graphs G_m (Fig. 3.2) from T_m as follows: first replace the root by the graph I_0, then all the nodes on a particular level are replaced by either I_1 or I_2 alternately (cf. Fig. 3.2). Now we add edges; suppose we are at a 'node' which is labeled I_i and the left child and right child are labeled I_j, we add an edge between v^i_ℓ and v^j_p in the

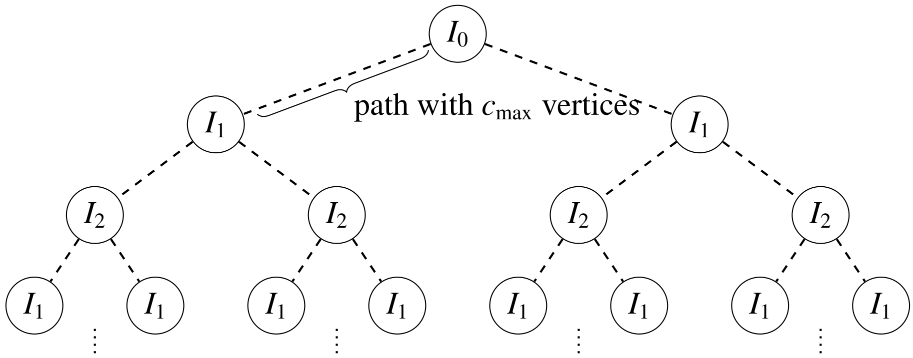

Figure 3.2: The graph G_m.

left child, and an edge between v_r^i and v_p^j in the right child. Finally, to obtain G_m we expand each added edge into a simple path with $c_{\max}$ vertices on it (cf. Fig. 3.2). That is, a left-edge (or, right-edge) connection between two incomparable graphs in the tree looks like, $I_i(v_\ell^i)$ – path with $c_{\max}$ vertices – $(v_p^j)I_j$ (or, $I_i(v_r^i)$ – path with $c_{\max}$ vertices – $(v_p^j)I_j$).

Theorem 3.4.2. *Over any field, the family of homomorphism polynomials* (f_m)*, with* $f_m(\overline{Y}) = f_{G_m,H_m}(\overline{Y})$*, where*

- G_m *is defined as above (see Fig. 3.2), and*
- H_m *is an undirected complete graph on* $\mathsf{poly}(m)$*, say* m^{13}*, vertices,*

is complete for VP *under p-projections.*

Proof. Membership in VP follows from Theorem 3.3.3.

We proceed with the *hardness* proof. The idea, as before, is to obtain the VP-complete universal polynomial $\mathsf{f}_{\mathfrak{U}_n}$ as a projection of f_m. Our starting point is the graph J_n' in Subsection 3.4.1. The graph J_n' is constructed in such a way that the parse trees of $\mathfrak{U}_n$ are now in bijection with complete binary trees (Observation 3.4.1).

We transform J_n' using the set $I = \{I_0, I_1, I_2\}$. This is similar to the transformation we did to the balanced binary tree T_m. We replace each vertex by a graph in I; $root_L$ gets I_0 and the rest of the layers get I_1 or I_2 alternately (as in Fig. 3.2). Edge connections are made so that a left/right child is connected to its parent via the edge $(v_p^j, v_\ell^i)/(v_p^j, v_r^i)$. Finally we replace each edge connection by a path with $c_{\max}$ vertices on it (as in Fig. 3.2), to obtain the graph R_n. All edges of R_n are labeled 1, with the following exceptions: Every input node contains the same rigid graph I_i. It has a vertex v_p^i. Each path connection to other nodes has this vertex as its end point. Label such path edges that are incident on v_p^i by the label of the input gate.

Let $m := 2^{k(n)}$. The choice of $\mathsf{poly}(m)$ is such that $O(c_{\max} \cdot s_n^2) \leqslant \mathsf{poly}(m)$, where s_n is the size of $\mathfrak{U}_n$. Hence, m^{13} suffices for the choice of $\mathsf{poly}(m)$. As before, the $\overline{Y}$ variables are set to $\{0, 1, \bar{x}\}$ such that the non-zero variables pick out the graph R_n. From the Observation 3.4.1 it follows that for each parse tree T of $\mathfrak{U}_n$, there exists a homomorphism $\phi\colon G_{2^{k(n)}} \to R_n$ such that $\mathsf{mon}(\phi)$ is exactly equal to $\mathsf{mon}(T)$. Recall $\mathsf{mon}(\cdot)$ denotes the monomial associated with an object. We claim that these are the only valid homomorphisms from $G_{2^{k(n)}} \to R_n$. We observe the following properties of homomorphisms from $G_{2^{k(n)}} \to R_n$, from which the claim follows. In the following by a rigid-node-subgraph we mean a graph in $\{I_0, I_1, I_2\}$ that replaces a vertex.

(i) Any homomorphic image of a rigid-node-subgraph of $G_{2^{k(n)}}$ in R_n, cannot split across two mutually incomparable rigid-node-subgraphs in R_n. That is, there cannot be two vertices in a rigid subgraph of $G_{2^{k(n)}}$ such that one of them is mapped into a rigid subgraph say n_1, and the other one is mapped into another rigid subgraph say n_2. This follows because homomorphisms do not increase distance.

(ii) Because of (i), with each homomorphic image of a rigid node $g_i \in G_{2^{k(n)}}$, we can associate at most one rigid node of R_n, say n_i, such that the homomorphic image of g_i is a subgraph of n_i and the paths (corresponding to incident edges) emanating from it. But such a subgraph has a homomorphism to n_i itself: fold each hanging path into an edge and then map this edge into an edge within n_i. (For instance, let ρ be a path hanging off n_i and attached to n_i at u, and let v be any neighbour of u within n_i. Mapping vertices of ρ to u and v alternately preserves all edges and hence is a homomorphism.) Therefore, we note that in such a case we have a homomorphism from $g_i \to n_i$. By rigidity and mutual incomparability, g_i must be the same as n_i, and this folded-path homomorphism must be the identity map. The other scenario, where we cannot associate any n_i because g_i is mapped entirely within connecting paths, is not possible since it contradicts *non-bipartiteness* of mutually-incomparable graphs.

Root must be mapped to the root: The rigidity of I_0 and Property (ii) implies that $I_0 \in G_{2^{k(n)}}$ is mapped identically to I_0 in R_n.

Every level must be mapped within the same level: The children of I_0 in $G_{2^{k(n)}}$ are mapped to the children of the root while respecting left-right behaviour. Firstly, the left child cannot be mapped to the root because of incomparability of the graphs I_1 and I_0. Secondly, the left child cannot be mapped to the right child (or vice versa) even though they are the same graphs, because the minimum distance between the vertex in I_0 where the left path emanates and the right child is $c_{\max} + 1$ whereas the distance between the vertex in I_0 where the left path emanates and the left child is $c_{\max}$. So some vertex from the left child must be mapped into the path leading to the right child and hence the rest of the left child must be mapped into a proper subgraph of right child. But this contradicts rigidity of I_1. Continuing like this, we can show that every level must map within the same level and that the mapping within a level is correct.

This completes the proof of VP-completeness. □

Thus from the VP-completeness and the upper bound of Theorem 3.3.3 we obtain the following characterisation of VP.

Corollary 3.4.3. *Let (G_n) and (H_n) be p-families of graphs. Consider the family of homomorphism polynomials $f = (f_n)$, where $f_n(\overline{Y}) = f_{G_n,H_n}(\overline{Y})$. Then,*

- *If the sequence (G_n) has bounded tree-width, $f \in$ VP.*

- *Moreover, there exists an explicit p-family* (G_m) *of bounded tree-width graphs, and* H_m *is a complete graph on* $O(m^{13})$ *vertices, such that* (f_{G_m,H_m}) *is* VP*-hard, over any field, with respect to p-projections.*

We observe that our algorithm for circuit construction from Lemma 3.3.2, along with the above characterisation, gives a way to construct *skew* circuits of size $2^{O(\log^2 n)}$ for every family (h_n) in VP (see also [MP08]). Consider the VP-hard family (f_{G_m,H_m}) given by Corollary 3.4.3. Since $(h_n) \in \mathsf{VP}$, h_n is a projection of f_{G_m,H_m} where m is p-bounded in n. Thus an arithmetic circuit computing h_n can be obtained from the circuit of f_{G_m,H_m} via the projection. But, from Lemma 3.3.2, we know that f_{G_m,H_m} has a skew circuit of size $m^{O(pw(G_m))}$. Now using the fact that $pw(G_m) \leqslant O(tw(G_m) \cdot \log m)$, and m is $\mathsf{poly}(n)$, we have a skew circuit of size $n^{O(\log n)}$ computing h_n.

3.5 Completeness : VBP

We now turn our attention to showing that homomorphism polynomials are also rich enough to characterize computation by algebraic branching programs. Indeed, there are many natural polynomials that are known to be complete for VBP, most notably the determinant family Det_n, iterated matrix multiplication IMM, and matrix powering, etc. (see [Bür00a, Blä01, MP08].) It is believed that VBP characterises "efficient" computation in linear algebra. In this section, we establish that there exists a p-family (G_k) of undirected *bounded path-width* graphs such that the family $(f_{G_k,H_k}(\overline{Y}))$ is VBP-complete with respect to p-projections.

As before, we use rigid and mutually incomparable graphs in the construction of G_k. Let $I = \{I_1, I_2\}$ be a set of two connected, non-bipartite, rigid and mutually incomparable graphs. Arbitrarily pick vertices $u \in V(I_1)$ and $v \in V(I_2)$. Let $c_{I_i} = |V(I_i)|$, and $c_{max} = \max\{c_{I_1}, c_{I_2}\}$. Consider the sequence of graphs G_k (Fig. 3.3); for every k, there is a simple path with $(k-1) + 2c_{max}$ edges between a copy of I_1 and I_2. The path is between the

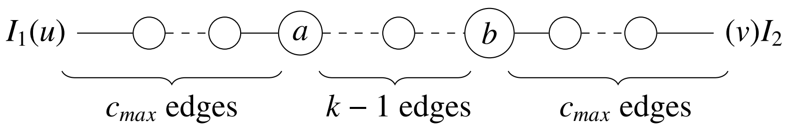

Figure 3.3: The graph G_k.

vertices $u \in V(I_1)$ and $v \in V(I_2)$. The path between vertices a and b in G_k contains $(k-1)$ edges.

In other words, connect I_1 and I_2 by stringing together a path with c_{max} edges between u and a, a path with $(k-1)$ edges between a and b, and a path with c_{max} edges between b and v.

Theorem 3.5.1. *Over any field, the family of homomorphism polynomials* (f_k)*, where*

- G_k *is defined as above (see Fig. 3.3),*
- H_k *is the undirected complete graph on* $O(k^2)$ *vertices,*
- $f_k(\overline{Y}) = f_{G_k,H_k}(\overline{Y})$,

is complete for VBP *with respect to p-projections.*

Proof. **Membership:** It follows from Theorem 3.3.3.

Hardness: Let $(g_n) \in$ VBP. Without loss of generality, we can assume that g_n is computable by a layered branching program of polynomial size such that the number of layers, ℓ, is more than the width of the algebraic branching program. We will show that g_n can be obtained as a projection of f_ℓ.

Let B'_n be the undirected graph underlying the layered branching program A_n for g_n. Let B_n be the following graph: $I_1(u) - (s)B'_n(t) - (v)I_2$, that is, $u \in I_1$ is connected to $s \in B'_n$ via a path with c_{max} edges and $t \in B'_n$ is connected to $v \in I_2$ via a path with c_{max} edges (cf. Fig. 3.3). The edges in B'_n inherits the weight from A_n, and the rest of the edges in B_n have weight 1.

Let us now consider f_ℓ when the variables on the edges of H_ℓ are instantiated to values in $\{0, 1\}$ or variables of g_n so that we obtain B_n as a subgraph of H_ℓ. We claim that a valid homomorphism from $\mathrm{G}_\ell \to B_n$ must satisfy the following properties:

(P1) I_1 in G_ℓ must be mapped to I_1 in B_n using the identity homomorphism,

(P2) I_2 in G_ℓ must be mapped to I_2 in B_n using the identity homomorphism.

Assuming the claim, it follows that homomorphisms from $\mathrm{G}_\ell \to B_n$ are in one-to-one correspondence with s-t paths in A_n. In particular, the vertex $a \in \mathrm{G}_\ell$ is mapped to the vertex s in B_n, and the vertex $b \in \mathrm{G}_\ell$ is mapped to the vertex t in B_n. Also, the monomial associated with a homomorphism and its corresponding path are the same. Therefore, we have,

$$f_{\mathrm{G}_\ell, B_n} = g_n.$$

Since ℓ is polynomially bounded, we obtain VBP-completeness of (f_k) over any field.

Let us now prove the claim. We first prove that a valid homomorphism from $\mathrm{G}_\ell \to B_n$ must satisfy the property (P1). There are three cases to consider.

- **Case 1:** *Some vertex of $V(I_1) \subseteq V(G_\ell)$ is mapped to u in B_n.* Since homomorphisms cannot increase distances between two vertices, we conclude that $V(I_1)$ must be mapped within the subgraph $I_1(u) - (a)$. Suppose further that some vertex on the $(u) - (a)$ path other than u is also in the homomorphic image of $V(I_1)$. Some neighbour of u in $V(I_1) \subseteq V(B_n)$, say u', must also be in the homomorphic image, since otherwise we have a homomorphism from the non-bipartite I_1 to a path, a contradiction. But note that $I_1(u) - (a)$ has a homomorphism to I_1: fold the $(u) - (a)$ path onto the edge $u - u'$ in I_1. Hence, composing the two homomorphisms we obtain a homomorphism from I_1 to I_1 which is not surjective. This contradicts the rigidity of I_1. So in fact the homomorphism must map $V(I_1)$ from G_ℓ entirely within I_1 from B_n, and by rigidity of I_1, this must be the identity map.

- **Case 2:** *Some vertex of $V(I_1) \subseteq V(G_\ell)$ is mapped to v in B_n.* Since homomorphisms cannot increase distances between two vertices, we conclude that $V(I_1)$ must be mapped within the subgraph $(b) - (v)I_2$. But note that $(b) - (v)I_2$ has a homomorphism to I_2 (fold the $(b) - (v)$ path onto any edge incident on v within I_2). Hence, composing the two homomorphisms, we obtain a homomorphism from I_1 to I_2. This is a contradiction, since I_1 and I_2 were incomparable graphs to start with.

- **Case 3:** *No vertex of $V(I_1) \subseteq V(G_\ell)$ is mapped to u or v in B_n.* Then $V(I_1) \subseteq V(\mathrm{G}_\ell)$ must be mapped entirely within one of the following disjoint regions of B_n: (i) $I_1 \setminus \{u\}$, (ii) bipartite graph between vertices u and v, and (iii) $I_2 \setminus \{v\}$. But then we contradict *rigidity of* I_1 in the first case, *non-bipartiteness of* I_1 in the second case, and *incomparability of* I_1 *and* I_2 in the last.

In a similar way, we could also prove that a valid homomorphism from $\mathrm{G}_\ell \rightarrow B_n$ must satisfy the property (P2). □

In the above proof, we crucially used incomparability of I_1 and I_2 to rule out flipping an undirected path. It turns out that over fields of characteristic not equal to 2, this is not crucial, since we can divide by 2. We show that if the characteristic of the underlying field is not equal to 2, then the sequence (G_k) in the preceding theorem can be replaced by a sequence of simple undirected cycles of appropriate length. In particular, we establish the following result.

Theorem 3.5.2. *Over fields of characteristic $\neq 2$, the family of homomorphism polynomials (f_k), $f_k = f_{G_k,H_k}$, where*

- *G_k is a simple undirected cycle of length $2k + 1$ and,*

- *H_k is an undirected complete graph on $(2k + 1)^2$ vertices,*

is complete for VBP *under p-projections.*

Proof. **Membership:** As before, it follows from Theorem 3.3.3.

Hardness: Let $(g_n) \in \mathsf{VBP}$. Without loss of generality, we can assume that g_n is computable by a layered branching program of polynomial size satisfying the following properties:

- The number of layers, $\ell \geqslant 3$, is odd; say $\ell = 2m + 1$. So every path from s to t in the branching program has exactly $2m$ edges.
- The number of layers is more than the width of the algebraic branching program,

Let us consider f_m when the variables on the edges of H_m have been set to 0, 1, or variables of g_n so that we obtain the undirected graph underlying the layered branching program A_n for g_n as a subgraph of H_m. Now change the weight of the (s, t) edge from 0 to weight y, where y is a new variable distinct from all the other variables of g_n. Call this modified graph B_m. Note that without the new edge, B_m would be bipartite, but with this edge it is not.

Let us understand the homomorphisms from G_m to B_m. Homomorphisms from a simple cycle C to a graph $\mathcal{G}$ are in one-to-one correspondence with closed walks of the same length in $\mathcal{G}$. Moreover, if the cycle C is of odd length, the closed walk must contain a simple odd cycle of at most the same length. Therefore, the only valid homomorphism from G_m to B_m are walks of length $\ell = 2m + 1$, and they all contain the edge (s, t) with weight y. But the cycles of length ℓ in B_m are in one-to-one correspondence with s-t paths in A_n. Each cycle contributes 2ℓ walks: we can start the walk at any of the ℓ vertices, and we can follow the directions from A_n or go against those directions. Thus we have,

$$f_{G_m,B_m} = (2(2m + 1)) \cdot y \cdot g_n = (2\ell) \cdot y \cdot g_n.$$

Let p be the characteristic of the underlying field. If $p = 0$, we substitute $y = (2\ell)^{-1}$ to obtain g_n. If $p > 2$, then 2ℓ has an inverse if and only if ℓ has an inverse. Since $\ell \geqslant 3$ is an odd number, either p does not divide ℓ or it does not divide $\ell + 2$. Hence, at least one of $\ell, \ell + 2$ has an inverse. Thus g_n is a projection of f_m or f_{m+1} depending on whether ℓ or $\ell + 2$ has an inverse in characteristic p.

Since $\ell = 2m + 1$ is p-bounded in n, we have therefore shown that (f_k) is VBP-complete with respect to p-projections over any field of characteristic other than 2. □

We also consider the directed variants of the homomorphism polynomial. Let **DirHom** denote the set of all directed homomorphisms between two directed graphs. We recall that directed homomorphisms preserve edges as well as their direction.

Theorem 3.5.3. *Over any field, the family of homomorphism polynomials (f_k), where*

- *G_k is a simple directed path $k + 1$ nodes $\langle u_1, u_2, \ldots, u_{k+1}\rangle$,*
- *H_k is the complete directed graph on $k(k + 1)$ nodes,*
- *$f_k(\overline{Y}) = f_{G_k,H_k,\mathbf{DirHom}}(\overline{Y})$,*

is complete for VBP *with respect to p-projections.*

Proof. The *membership* follows from Theorem 3.3.3.

Hardness: We reduce the iterated matrix multiplication family (IMM_n) to (f_k). IMM_n is the polynomial computed by an ABP with (1) a source node s, $n-1$ layers of n nodes each, and a target node t, (2) complete bipartite graphs between layers, and (3) distinct variables $\bar{x}$ on all edges. We will denote this ABP by B_n. (IMM_n) is known to be VBP-complete with respect to p-projections.

Let us consider f_n when the variables on the edges of H_n have been set to 0 or $\bar{x}$ so that we obtain the layered branching program B_n for IMM_n as a subgraph of H_n.

For every s-t path ρ in B_n, there is a homomorphism ϕ from G_n to B_n such that $\mathsf{mon}(\phi) = \mathsf{mon}(\rho)$. Conversely, for any homomorphism ϕ from G_n to B_n, ϕ must map G_n to a proper path between s and t. This follows from two facts: (i) directed homomorphisms from a directed path are in one-to-one correspondence with directed walks of the same length in the target graph, and (ii) acyclicity of B_n (which forces that paths of length n in B_n exist only between s and t). So $\mathsf{mon}(\phi)$ is in fact $\mathsf{mon}(\rho)$ for some s-t path ρ. Hence IMM_n is the projection of f_n. □

Theorem 3.5.4. *Over any field, the family of homomorphism polynomials (f_k), where*

- *G_k is a simple directed cycle on k nodes $\langle u_1, u_2, \ldots, u_k, u_1\rangle$,*
- *H_k is the complete directed graph on k nodes,*
- *$f_k(\overline{Y}) = f_{G_k,H_k,\mathbf{DirHom}}(\overline{Y})$,*

is complete for VBP *with respect to p-projections.*

Proof. Again the *membership* follows from Theorem 3.3.3. We only sketch the *hardness* proof here.

Consider the family of polynomials (F_n) such that $F_n = \mathsf{Tr}(X^n)$, where X is an $n \times n$ symbolic matrix, and Tr denotes the trace of a matrix. It is known that the family (F_n) is VBP-complete with respect to p-projections [MP08].

We claim that F_n is a projection of f_n. The claim easily follows from the observation that directed homomorphisms from a directed cycle are in one-to-one correspondence with directed closed walks of the same length in the target graph. □

3.6 Completeness : VNP

In this section, we present a homomorphism polynomial that is complete for VNP with respect to p-projections. For each $n \in \mathbb{N}$, let $\mathcal{I}_n := \{I_{n1}, I_{n2}, \ldots, I_{nn}\}$ be a set of n rigid and mutually incomparable graphs. If the subscript n is clear from the context, we will drop

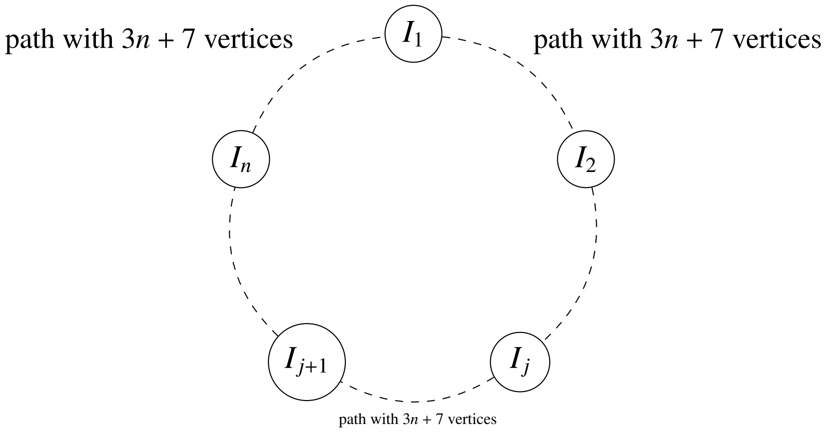

Figure 3.4: The Graph G_n.

it and write $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$. We can further assume that for all $j \in [n]$, I_j is defined on $\Theta(n)$ vertices, in fact $3n + 7$ vertices, and its tree-width is also $\Theta(n)$ (see Section 3.7). For each I_j, we mark two distinct vertices t_j and s_j in its vertex set. Consider the sequence of graphs G_n (see Fig. 3.4). In words, place I_1 to I_n on an n-cycle, connect the big nodes with the edges from the set $C := \left\{(s_j, t_{j+1}) \mid j \in [n-1]\right\} \cup \{(s_n, t_1)\}$, and finally, to obtain the graph G_n, stretch each edge in C into a path with $3n + 7$ vertices on it.

Theorem 3.6.1. *Over any field, the family of homomorphism polynomials (f_n), where*

- *G_n is defined as above (see Fig. 3.4),*
- *H_n is the undirected complete graph on $O(n^4)$ vertices,*
- *$f_n(\overline{Y}) = f_{G_n,H_n}(\overline{Y})$,*

is complete for VNP *with respect to p-projections.*

Proof. Membership in VNP follows from Proposition 3.3.1. To establish *hardness* we will show that the Hamiltonian cycle family (HC_n) is a p-projection of (f_n). Recall that HC_n is defined as in Section 2.5. We now construct a graph UK_n on $O(n^4)$ vertices such that $f_{G_n,UK_n} = \mathsf{HC}_n$. A suitable projection can then restrict H_n to UK_n, showing that HC_n is a projection of f_n.

Consider a copy of I_1, and for each $j \in \{2, \ldots, n\}$, $n-1$ copies of I_j, denoted I_j^i for $i \in \{2, \ldots, n\}$. Let K_n denote a complete directed graph on n vertices $\{v_i \mid i \in [n]\}$.

We will modify K_n to obtain UK_n. We first replace the vertices of K_n as follows: replace v_1 with I_1, and for $i \in \{2, \ldots, n\}$, replace v_i with the set $\{I_j^i \mid 2 \leqslant j \leqslant n\}$. Intuitively by such a replacement we isolate the vertex v_1, and thus make it always the first vertex in a Hamiltonian cycle. This further helps in counting each Hamiltonian cycle exactly once.

Now we add the *connector* edges as follows.

For each edge $\langle v_1, v_i \rangle$ such that $i \neq 1$, we add the edge (s_1, t_2^i) with weight $X_{1,i}$, where s_1 is the marked vertex in I_1, and t_2^i is the marked vertex in I_2^i. Intuitively, using this edge in homomorphisms correspond to using the edge $\langle v_1, v_i \rangle$ in a Hamiltonian cycle.

For each edge $\langle v_i, v_1 \rangle$ such that $i \neq 1$, add (s_n^i, t_1) with weight $X_{i,1}$, where s_n^i is the marked vertex in I_n^i, and t_1 is the marked vertex in I_1. As before, using this edge in homomorphisms correspond to using the edge $\langle v_i, v_1 \rangle$ in a Hamiltonian cycle.

For each edge $\langle v_i, v_j \rangle$ such that $i \neq j$ and $1 \notin \{i, j\}$, add the following edges $\{(s_k^i, t_{k+1}^j) \mid k \in \{2, \ldots, n-1\}\}$. Moreover, they all have the same weight $X_{i,j}$. As before, s_k^i is the marked vertex in I_k^i and t_{k+1}^j is the marked vertex in I_{k+1}^j. Intuitively, using the edge (s_k^i, t_{k+1}^j) in homomorphisms correspond to using the edge $\langle v_i, v_j \rangle$ in a Hamiltonian cycle such that the vertex v_i is in the k-th position and v_j is in the $(k+1)$-th position.

Now we stretch the connector edges into a path with $3n+7$ vertices on it. Put the label of the connector edge onto the middle edge of this path. Rest of the edges in the path have weight 1. We denote this graph with UK_n. Clearly UK_n is defined on $O(n^4)$ vertices.

We now prove our claim that $\mathsf{HC}_n = f_{G_n, UK_n}$. To prove the claim it suffices to show that homomorphisms from G_n to UK_n are in one-to-one correspondence with the Hamiltonian cycles in K_n. It easily follows that every Hamiltonian cycle gives a homomorphic mapping of G_n into UK_n by following the cycle (based on the intuition described before). For example, if $\langle v_1, v_{k_1}, \ldots v_{k_{n-1}} \rangle$ is a Hamiltonian cycle in K_n, then the homomorphic map of

G_n into UK_n is given as follows: I_1 in G_n maps to I_1 in UK_n using identity mapping, then I_2 in G_n is mapped to $I_2^{k_1}$ in UK_n using identity mapping, and, in general, I_i in G_n is mapped to $I_i^{k_{i-1}}$ in UK_n using identity mapping. For the reverse direction, we use (*i*) the rigidity and incomparability of the set $\mathcal{I}$, and (*ii*) the fact that homomorphisms cannot increase distance. Using these two facts we first argue that each rigid node in G_n (from the set $\mathcal{I}$) must map identically to one of its copy in UK_n. We can further argue that no two rigid nodes in G_n can be mapped into the set associated with a single vertex in UK_n. That is, distinct I_i and I_j in G_n can not be mapped simultaneously to I_i^k and I_j^k for any $k \in \{2, \ldots, n\}$. Thus we have shown that a homomorphism from G_n to UK_n necessarily picks out a n-cycle in K_n. Now by the fact there is only one copy of I_1 in UK_n it follows that I_1 in G_n must be mapped to I_1 in UK_n using the identity mapping. This uniquely defines the direction of the n-cycle, and hence each cycle is counted exactly once. $\square$

Based on the discussion, so far, we can say that VNP is characterised by the homomorphism polynomials where the p-family of graphs (G_n) is such that tree-width of G_n is $\Theta(n)$. VP is characterised by the homomorphism polynomials where the family (G_n) have bounded tree-width (independent of n). Furthermore, VBP is characterised by the homomorphism polynomials where the sequence (G_n) have bounded path-width. This raises an interesting question.

> *What is the complexity of homomorphism polynomials that are defined on a family G_n such that G_n has tree-width* $o(n)$*?*

In [HY11], it was shown that most polynomials in n variables and degree n with zero-one coefficients require circuits of size at least $\Omega(2^n)$. With such an evidence, it wouldn't be far fetched to conjecture that complete families are the one that require exponential complexity. More precisely, consider the following hypothesis which is stronger than Valiant's hypothesis VP $\neq$ VNP.

(H) For any VNP-complete family (f_n), there exist an $\epsilon \in (0, \frac{1}{2}]$ such that $\mathsf{size}_\mathsf{c}(f_n) \geqslant 2^{n^{(1/2)+\epsilon}}$ for infinitely many n.

Recall the family Clique^k defined in Section 2.5, where

$$\mathsf{Clique}_n^k := \sum_{\substack{S\subseteq[n]\\|S|=k}} \prod_{\substack{i,j\in S\\i<j}} x_{i,j}.$$

It enumerates k-sized cliques in an n-vertex graph.

Set $k = \log n$. By definition, it follows that $\mathsf{Clique}_n^{\log n}$ is computable by an arithmetic circuit of size $n^{O(\log n)}$. Consequently, if $\mathsf{Clique}^{\log n}$ is VNP-complete then all families in VNP will have $n^{O(\log n)}$-sized circuits computing them. This contradicts the hypothesis **(H)**.

Similarly, if $\mathsf{Clique}_n^{\log n}$ is in VP, then using Lemma 1 from [FK97], it follows that Clique_n^k has a circuit of size $n^{O(\sqrt{k})}$ for any k. (Feige and Kilian [FK97] reduced the decision version of Clique^k to the decision version of $\mathsf{Clique}^{\log n}$ in $n^{O(\sqrt{k})}$ time assuming that the decision version of $\mathsf{Clique}^{\log n}$ is solvable in P.) In particular, $\mathsf{Clique}_n^{n/2}$ is computable by circuits of size $2^{O(\sqrt{n}\log n)}$. But $\mathsf{Clique}^{n/2}$ is VNP-complete, and hence we reach a contradiction to the hypothesis **(H)**.

From these observations, we obtain the following proposition.

Proposition 3.6.2. *Under the hypothesis* **(H)***, over any field,* $\mathsf{Clique}^{\log n}$ *is neither in* VP*, nor* VNP*-hard.*

We call such polynomial families that belong to VNP, but are not in VP and not VNP-hard, VNP-intermediate. Although the above intermediate result Proposition 3.6.2 is established under too strict a hypothesis, the purpose is to motivate the study of "natural" VNP-intermediate families. We will continue our discussion on VNP-intermediate families in Chapter 4. We now end this chapter with a description of an explicit family of *rigid* and *mutually incomparable* graphs. These can be used in the hardness proofs in Section 3.4.2, Section 3.5, and Section 3.6

3.7 Rigid and Incomparable graphs

We describe a sequence of set of rigid and mutually incomparable graphs given by Hell and Nešetřil (Exercise 6, Chapter 4, [HN04]).

Let $1 \leqslant \ell \leqslant n$. Consider the following graph $H(n, \ell)$: the vertex set is $\{1, 2, \ldots, 3n + 7\}$, and the edges are $(1, 3n + 7), (1, n + 4 + \ell)$ and all (i, j) with $1 \leqslant |i - j| \leqslant n + 1$.

Lemma 3.7.1. *The graph $H(n, \ell)$ as defined above satisfy the following properties:*

- *Each $H(n, \ell)$ is rigid.*
- *There is no homomorphism $H(n, \ell) \to H(n, \ell')$ for $\ell \neq \ell'$.*

We illustrate the proof of Lemma 3.7.1 by showing that the graphs $H(3, 1), H(3, 2)$, and $H(3, 3)$ (see Fig. 3.5) are rigid and pairwise-incomparable. For the purpose of proof we will partition the vertices into three classes, namely Red, Blue, and Green (cf. Fig. 3.5). The vertices of $H(3, i)$, for $1 \leqslant i \leqslant 3$, are partitioned as follows: the vertex $(7 + i)$ is in the Red set, the vertices 1 and 16 are in the Blue set, and the rest of the vertices are in the Green set.

A graph H is *asymmetric* if the only *automorphism* (isomorphism from H to itself) is the identity. A graph H is a *core* if every *endomorphism* (homomorphism from H to itself) is an isomorphism (and hence an automorphism). A graph H is rigid if the only endomorphism is the identity. H is rigid if and only if it is an asymmetric core.

Let χ_H denote the chromatic number of H, that is, the least k such that some map from $V(H)$ to the set of colours $[k]$ gives all adjacent vertices distinct colours. If there is a homomorphism from G to H, then the definition of homomorphism implies that $\chi(G) \leqslant$

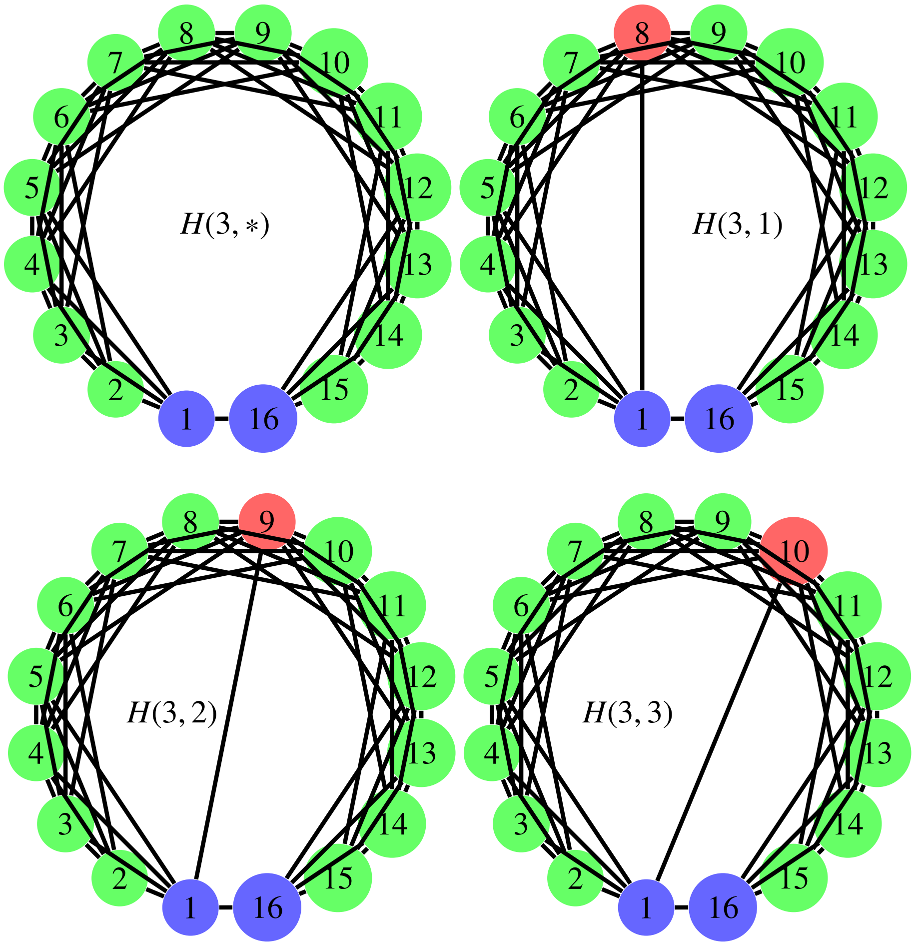

Figure 3.5: $H(3,1)$, $H(3,2)$, $H(3,3)$: three rigid pairwise-incomparable graphs.

$\chi(H)$. Hence, if we define *vertex-criticality* saying that that H is vertex-critical if for every $u \in V(H)$, $\chi_{H\setminus\{v\}} < \chi_H$, then it follows that every vertex-critical graph is a core.

Claim 3.7.1. *Each graph in* $\{H(3,*), H(3,1), H(3,2), H(3,3)\}$ *is a core.*

Claim 3.7.2. *Each graph in* $\{H(3,1), H(3,2), H(3,3)\}$ *is asymmetric.*

Hence, each $H(3,i)$ for $i \in [3]$ is rigid.

Claim 3.7.3. *The graphs in* $\{H(3,1), H(3,2), H(3,3)\}$ *are pairwise incomparable; for* $i \neq j$*, there is no homomorphism from* $H(3,i)$ *to* $H(3,j)$*.*

Proof of Claim 3.7.1. We show that $H(G,*)$ (and hence also each $H(3,i)$) is not 5-colourable, while for every $u \in [16]$, each $H(3,i) \setminus \{u\}$ is 5-colourable. Hence all 4 graphs are 6-chromatic vertex-critical.

Non-5-colourability: The vertices 1 to 5 form a clique and must get distinct colours, say 1 to 5. Now there is a unique way of extending the colouring sequentially to 6,7,8, ... But this assigns colour 1 to 16, and 1 and 16 are neighbours. So no 5-colouring is possible.

5-colourability: Consider $H(3,i) \setminus \{u\}$. Colour node j with colour j mod 5 if $j < u$, with colour $j-1$ mod 5 if $j > u$. This satisfies all edge constraints: For a black edge (j,k), $1 \leqslant |j-k| \leqslant 4$, so if both j and k are present, then their colours are distinct even if $j < u < k$. If the blue-red edge is present, note that the red vertex gets colour 2,3,4,or 5, while vertex 1 always gets colour 1.

□

Proof of Claim 3.7.2. Since isomorphisms must preserve degrees vertex-wise, consider the degrees of vertices in the graphs. First, group the vertices of $H(3,*)$ by degree.

degree 5: $\{1, 2, 15, 16\}$

degree 6: $\{3, 14\}$

degree 7: $\{4, 13\}$

degree 8: $\{5, 6, 7, 8, 9, 10, 11, 12\}$.

Similarly, group the vertices of $H(3, i)$ by degree.

degree 5: $\{2, 15, 16\}$

degree 6: $\{1, 3, 14\}$

degree 7: $\{4, 13\}$

degree 8: $\{5, 6, 7, 8, 9, 10, 11, 12\}$ \ {the red node 7+i}

degree 9: the red node $7 + i$

Consider an automorphism f on $H(3, 1)$. Since only vertex 8 has degree 9, f must map 8 to 8. Vertex 1 is the only neighbour of 8 with degree 6, so f must map 1 to 1. Vertex 1 has two degree-5 neighbours, 2 and 16, but 16 has another degree-5 neighbour 15 while 2 does not have any degree-5 neighbour, so f cannot swap these degree-5 neighbours of 1. So f maps 2 to 2 and 16 to 16. Proceeding this way based on degree, we see that f must in fact fix every vertex.

An identical argument works for $H(3, 2)$. For $H(3, 3)$, one additional twist: the red vertex 10 gets mapped to 10. Now 10 has two degree-6 neighbours, 1 and 14. Can f map 1 to 14? But 1 has a degree-6 neighbour 3, while 14 has no degree-6 neighbour. So f cannot swap 1 and 14.

□

Proof of Claim 3.7.3. Suppose to the contrary that $f : V_1 \to V_2$ is a homomorphism from $H(3, 1)$ to $H(3, 2)$ (the argument is similar for other pairs). If f is not surjective, then by vertex-criticality, $H(3, 1)$ has a homomorphism to a 5-colourable graph, but $\chi(H(3, 1)) = 6$, a contradiction. So f must be surjective.

Furthermore, f must induce a bijection between the edges of $H(3, 1)$ and $H(3, 2)$. If it didn't, then two edges of $H(3, 1)$ are mapped to the same edge of $H(3, 2)$. This implies that two vertices of $H(3, 1)$ are mapped to the same vertex of $H(3, 2)$, violating surjectivity.

Thus the vertex degrees must be preserved exactly: for each $u \in V_1$, the degree of u in $H(3, 1)$ is the same as the degree of $f(u)$ in $H(3, 2)$.

Since the red vertices are the only vertices with degree 9, f must map the red vertex of $H(3,1)$, vertex 8, to the red vertex of $H(3,2)$, vertex 9. Now use the argument as used in Claim 3.7.2 to extend this mapping. f must map 1 to 1, 2 to 2, and so on. We thus reach the conclusion that f must map 8 to 8, contradicting $f(8) = 9$. Hence no such map f is possible.

□

3.8 Conclusion

In this chapter, we studied families of polynomials defined using graph homomorphisms, and characterised the algebraic classes VBP, VP, and VNP. We also provide a first instance of natural families of polynomials that are VP-complete with respect to p-projections. Our work raises further interesting questions on the complexity of the homomorphism polynomials. In particular,

- *What is the complexity of homomorphism polynomials that are defined on a family G_n such that G_n has tree-width* $o(n)$*?* (Also, see the discussion around Proposition 3.6.2.)
- A striking aspect of Perm being VNP-complete is that the underlying decision problem, in fact even the search problem, is in P. This helped in establishing VNP-completeness of a host of other polynomials by reduction from the Perm family. *Can we use the homomorphism polynomials to unearth new natural families that are* VP*-complete?*
- Consider the partial order over p-families, under the relation p-projection. *Can we characterise the degrees of p-families, in the aforementioned poset, using the homomorphism polynomials?*

Chapter 4

Polynomials with intermediate complexity

4.1 Introduction

A plethora of natural problems are either known to be NP-complete or are in P (see, for example, [GJ79]). This raised a speculation about the possibility that all problems are either NP-complete or in P. This view was quickly proven wrong by Ladner [Lad75]. He showed that assuming $\mathsf{P} \neq \mathsf{NP}$, there exists a language L in $\mathsf{NP} \setminus \mathsf{P}$ such that L is not NP-complete (even under oracle reductions).

Inspired from such classical results, Bürgisser [Bür99] proved, among other things, that over any field, if Valiant's hypothesis (i.e. $\mathsf{VP} \neq \mathsf{VNP}$) is true, then there is a p-family in VNP which is neither in VP nor VNP-complete with respect to c-reductions. We call such a polynomial family VNP-intermediate, that is, it is (1) in VNP, (2) not VNP-complete, and (3) not in VP. Bürgisser [Bür99] further showed that, over finite fields of characteristic p, a *specific* family of polynomials is VNP-intermediate provided $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$. He also showed that the condition $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$ is met if the polynomial hierarchy

PH does not collapse to the second level. Hence a very reasonable assumption.

At an intuitive level, Bürgisser's intermediate polynomial family enumerates *cuts* in a graph. This is a remarkable result, when compared with the classical P-NP setting or the BSS-model. The existence of problems with intermediate complexity has been established in the latter settings. But these problems seem highly unnatural owing to the involved "diagonalization" arguments that are used in their construction. In other words, their definitions are not motivated by an underlying combinatorial problem but guided by the needs of the proof and, hence, seem artificial. The question of whether there are other naturally-defined VNP-intermediate polynomial families, recently highlighted again in [Gro15], was left open by Bürgisser [Bür00a].

In this chapter we establish a list of new natural VNP-intermediate polynomial families. The definitions of these families are motivated by basic (combinatorial) NP-complete problems that are complete under *parsimonious* reductions.

We mention some basics in Section 4.2. We then define the new polynomial families in Section 4.3, and also establish their intermediate complexity.

4.2 Preliminaries

Let P/poly denote the class of languages decidable by polynomial-sized Boolean circuit families. A function $\phi : \{0,1\}^* \to \mathbb{N}$ is in #P if there exists a polynomial p and a polynomial time deterministic Turing machine M such that for all $x \in \{0,1\}^*$, $\phi(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}|$. For a prime p, define

$$\#_p\mathsf{P} = \{\psi : \{0,1\}^* \to \mathbb{F}_p \mid \psi(x) = \phi(x) \bmod p \text{ for some } \phi \in \#\mathsf{P}\},$$

$$\mathsf{Mod}_p\mathsf{P} = \{L \subseteq \{0,1\}^* \mid \text{ for some } \phi \in \#\mathsf{P}, x \in L \iff \phi(x) \equiv 1 \bmod p\}$$

It is easy to see that if $\phi : \{0,1\}^* \to \mathbb{N}$ is #P-complete with respect to parsimonious reductions (that is, for every $\psi \in \#P$, there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$, $\psi(x) = \phi(f(x))$), then the language $L = \{x \mid \phi(x) \equiv 1 \bmod p\}$ is $\mathsf{Mod}_p\mathsf{P}$-complete with respect to many-one reductions.

4.3 Intermediate polynomials

From Bürgisser's proof [Bür99] that the *cut enumerator* family is VNP-intermediate, we can abstract out the following strategy to establish VNP-intermediate families.

Find an explicit polynomial family $h = (h_n)$ satisfying the following properties.

M: Membership. The family is in VNP.

E: Ease. Over a field $\mathbb{F}_q$ of size q and characteristic p, h can be evaluated in P. Thus if h is VNP-hard, then we can efficiently compute #P-hard functions, modulo p.

H: Hardness. The monomials of h encode solutions to a problem that is #P-hard via parsimonious reductions. Thus if h is in VP, then the number of solutions, modulo p, can be extracted using coefficient computation.

Then, unless $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$ (which in turn implies that PH collapses to the second level, [Bür00a, KL82]), h is VNP-intermediate.

We will demonstrate the above proof strategy on the families of polynomials that we define. As hinted in the introduction, these families are based on basic NP-complete problems that are complete under parsimonious reductions. We now describe them in detail. Two of these, $\mathsf{Sat}^{\mathsf{q}}$ and $\mathsf{Clow}^{\mathsf{q}}$, were defined earlier in Section 2.5.3. However for ease of reading we repeat the definitions here.

(1) The *satisfiability* polynomial $\mathsf{Sat}^{\mathsf{q}} = (\mathsf{Sat}^{\mathsf{q}}{}_n)$: For each n, let Cl_n denote the set of all possible clauses of size 3 over $2n$ literals. There are n variables $\tilde{X} = \{X_i\}_{i=1}^n$, and also $8n^3$

clause-variables $\tilde{Y} = \{Y_c\}_{c \in \mathsf{Cl}_n}$, one for each 3-clause c.

$$\mathsf{Sat^q}_n := \sum_{a \in \{0,1\}^n} \left(\prod_{i \in [n]: a_i = 1} X_i^{q-1} \right) \left(\prod_{\substack{c \in \mathsf{Cl}_n \\ a \text{ satisfies } c}} Y_c^{q-1} \right).$$

For the next three polynomials, we consider the complete graph G_n on n nodes, and we have the set of variables $\tilde{X} = \{X_e\}_{e \in E_n}$ and $\tilde{Y} = \{Y_v\}_{v \in V_n}$.

(2) The *vertex cover* polynomial $\mathsf{VC^q} = (\mathsf{VC^q}_n)$:

$$\mathsf{VC^q}_n := \sum_{S \subseteq V_n} \left(\prod_{e \in E_n :\ e \text{ is incident on } S} X_e^{q-1} \right) \left(\prod_{v \in S} Y_v^{q-1} \right).$$

(3) The *clique/independent set* polynomial $\mathsf{CIS^q} = (\mathsf{CIS^q}_n)$:

$$\mathsf{CIS^q}_n := \sum_{T \subseteq E_n} \left(\prod_{e \in T} X_e^{q-1} \right) \left(\prod_{v \text{ incident on } T} Y_v^{q-1} \right).$$

(4) The *clow* polynomial $\mathsf{Clow^q} = (\mathsf{Clow^q}_n)$: A clow in an n-vertex graph is a closed walk of length exactly n, in which the minimum numbered vertex (called the head) appears exactly once.

$$\mathsf{Clow^q}_n := \sum_{w:\ \text{clow of length } n} \left(\prod_{e:\ \text{edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v:\ \text{vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

If an edge e is used k times in a clow, it contributes $X_e^{k(q-1)}$ to the monomial. But a vertex v contributes only Y_v^{q-1} even if it appears more than once. More precisely,

$$\mathsf{Clow^q}_n := \sum_{\substack{w = \langle v_0, v_1, \ldots, v_{n-1} \rangle: \\ \forall j > 0,\ v_0 < v_j}} \left(\prod_{i \in [n]} X_{(v_{i-1}, v_{i \bmod n})}^{q-1} \right) \left(\prod_{v \in \{v_0, v_1, \ldots, v_{n-1}\}} Y_v^{q-1} \right).$$

(5) The *3D-matching* polynomial $\mathsf{3DM^q} = (\mathsf{3DM^q}_n)$: Consider the complete tripartite

hyper-graph, where each part in the partition (A_n, B_n, C_n) contain n nodes, and each hyperedge has exactly one node from each part. We have variables X_e for hyperedge e and Y_v for node v.

$$\mathsf{3DM^q}_n := \sum_{M \subseteq A_n \times B_n \times C_n} \left(\prod_{e \in M} X_e^{q-1} \right) \left(\prod_{\substack{v \in M \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

We show that if $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, then all five polynomials defined above are VNP-intermediate.

Observe that in the polynomials above, the combinatorial object of interest is encoded in a somewhat non-standard way. For instance, the clique-independent set polynomial $\mathsf{CIS^q}$ has monomials where the X_e variables correspond to any subset of edges, not just subsets arising from cliques. The idea is that padding a polynomial with "useless monomials" can make it easier to compute, hence avoiding VNP-completeness. At the same time, the padding is carefully chosen so that the interesting objects can still be retrieved with some overhead. For instance, the Y_u variables in the monomials of $\mathsf{CIS^q}$ allow us to distinguish between useful and useless monomials. Hence the polynomial does not become so easy to compute that it lies in VP. Thus the major obstacle in establishing VNP-intermediate families is in identifying the right amount of padding to achieve both these goals.

Theorem 4.3.1. *Over a finite field $\mathbb{F}_q$ of characteristic p, the polynomial families* $\mathsf{Sat^q}$, $\mathsf{VC^q}$, $\mathsf{CIS^q}$, $\mathsf{Clow^q}$, *and* $\mathsf{3DM^q}$, *are in* VNP. *Further, if* $\mathsf{Mod}_p\mathsf{P} \not\subseteq \mathsf{P/poly}$, *then they are all* VNP*-intermediate; that is, neither in* VP *nor* VNP*-hard with respect to c-reductions.*

Proof. **(M)** An easy way to see membership in VNP is to use Valiant's criterion (Proposition 2.2.12), that is, the coefficient of any monomial can be computed efficiently, hence the polynomial is in VNP. This establishes membership for all families.

We first illustrate the rest of the proof by showing that the polynomial $\mathsf{Sat^q}$ satisfies the properties **(H)**, **(E)**.

(H) Assume $(\mathsf{Sat}^{\mathsf{q}}{}_n)$ is in VP, via polynomial-sized circuit family $\{C_n\}_{n\geqslant 1}$. We will use C_n to give a $\mathsf{P/poly}$ upper bound for computing the number of satisfying assignments of a 3-CNF formula, modulo p. Since this question is complete for $\mathsf{Mod}_p\mathsf{P}$, the upper bound implies $\mathsf{Mod}_p\mathsf{P}$ is in $\mathsf{P/poly}$.

Given an instance ϕ of 3-SAT, with n variables and m clauses, consider the projection of $\mathsf{Sat}^{\mathsf{q}}{}_n$ obtained by setting all Y_c for $c \in \phi$ to t, and all other variables to 1. This gives the polynomial $\mathsf{Sat}^{\mathsf{q}}\phi(t) = \sum_{j=1}^{m} d_j t^{j(q-1)}$ where d_j is the number of assignments (modulo p) that satisfy exactly j clauses in ϕ. Our goal is to compute d_m.

We convert the circuit C into a circuit D that compute elements of $\mathbb{F}_q[t]$ by explicitly giving their coefficient vectors, so that we can pull out the desired coefficient. (Note that after the projection described above, C works over the polynomial ring $\mathbb{F}_q[t]$.) Since the polynomial computed by C is of degree $m(q-1)$, we need to compute the coefficients of all intermediate polynomials too only upto degree $m(q-1)$. Replacing $+$ by gates performing coordinate-wise addition, $\times$ by a sub-circuit performing (truncated) convolution, and supplying appropriate coefficient vectors at the leaves gives the desired circuit. Since the number of clauses, m, is polynomial in n, the circuit D is also of polynomial size. Given the description of C as advice, the circuit D can be evaluated in P, giving a $\mathsf{P/poly}$ algorithm for computing #3-$\mathsf{SAT}(\phi)$ mod p. Hence $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$.

(E) Consider an assignment to $\tilde{X}$ and $\tilde{Y}$ variables in $\mathbb{F}_q$. Since all exponents are multiples of $(q-1)$, it suffices to consider 0/1 assignments to $\tilde{X}$ and $\tilde{Y}$. Each assignment a contributes 0 or 1 to the final value; call it a contributing assignment if it contributes 1. So we just need to count the number of contributing assignments. An assignment a is contributing exactly when $\forall i \in [n]$, $X_i = 0 \implies a_i = 0$, and $\forall c \in \mathsf{Cl}_n$, $Y_c = 0 \implies a$ does not satisfy c. These two conditions, together with the values of the X and Y variables, constrain many bits of a contributing assignment; an inspection reveals how many (and which) bits are so constrained. If any bit is constrained in conflicting ways (for example, $X_i = 0$, and $Y_c = 0$ for some clause c containing the literal $\bar{x}_i$), then no assignment is contributing (either $a_i = 1$ and the X part becomes zero due to $X_i^{a_i}$, or $a_i = 0$ and the Y part becomes zero due to Y_c). Otherwise, some bits of a potentially contributing assignment are constrained by X and Y, and the remaining bits can be set in any way. Hence the total sum is precisely $2^{(\text{\# unconstrained bits})} \bmod p$.

Now assume $\mathsf{Sat^q}$ is VNP-hard. Let L be any language in $\mathsf{Mod}_p\mathsf{P}$, witnessed via #P-function f. (That is, $x \in L \iff f(x) \equiv 1 \bmod p$.) By the results of [Bür00b] (see also [Bür00a]), there exists a p-family $r = (r_n) \in \mathsf{VNP}_{\mathbb{F}_p}$ such that $\forall n$, $\forall x \in \{0,1\}^n$, $r_n(x) = f(x) \bmod p$. By assumption, there is a c-reduction from r to $\mathsf{Sat^q}$. We use the oracle circuits from this reduction to decide instances of L. On input x, the advice is the circuit C of appropriate size reducing r to $\mathsf{Sat^q}$. We evaluate this circuit bottom-up. At the leaves, the values are known. At $+$ and $\times$ gates, we perform these operations in $\mathbb{F}_q$. At an oracle gate, the paragraph above tells us how to evaluate the gate. So the circuit can be evaluated in polynomial time, showing that L is in $\mathsf{P/poly}$. Thus $\mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{P/poly}$.

For the other four families, it suffices to show the following, since the rest is identical as for $\mathsf{Sat^q}$.

H′. The monomials of h encode solutions to a problem that is #P-hard via parsimonious reductions.

E′. Over $\mathbb{F}_q$, h can be evaluated in P.

We describe this for the polynomial families one by one.

The *vertex cover* polynomial $\mathsf{VC^q} = (\mathsf{VC^q}_n)$:

$$\mathsf{VC^q}_n := \sum_{S \subseteq V_n} \left(\prod_{e \in E_n :\, e \text{ is incident on } S} X_e^{q-1} \right) \left(\prod_{v \in S} Y_v^{q-1} \right).$$

(H′) Given an instance of vertex cover $A = (V(A), E(A))$ such that $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{VC^q}_n$ encodes the number of solutions of instance A. Consider the following projection of $\mathsf{VC^q}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. Thus, we have

$$\mathsf{VC^q}_n(z,t) = \sum_{S \subseteq V_n} z^{(\text{\# edges incident on } S)(q-1)} t^{|S|(q-1)}.$$

Hence, it follows that the number of vertex cover of size k, modulo p, is the coefficient of $z^{m(q-1)} t^{k(q-1)}$ in $\mathsf{VC^q}_n(z,t)$.

(E′) Consider the weighted graph given by the values of $\tilde{X}$ and $\tilde{Y}$ variables. Each subset $S \subseteq V_n$ contributes 0 or 1 to the total. A subset $S \subseteq V_n$ contributes 1 to $\mathsf{VC^q}_n$ if and only if every vertex in S has non-zero weight, and every edge incident on each vertex in S has non-zero weight. That is, S is a subset of full-degree vertices. Therefore, the total sum is $2^{(\text{\# full-degree vertices})} \bmod p$.

The *clique/independent set* polynomial $\mathsf{CIS^q} = (\mathsf{CIS^q}_n)$:

$$\mathsf{CIS^q}_n := \sum_{T \subseteq E_n} \left(\prod_{e \in T} X_e^{q-1} \right) \left(\prod_{v \text{ incident on } T} Y_v^{q-1} \right).$$

(H′) Given an instance of clique $A = (V(A), E(A))$ such that $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{CIS^q}_n$ encodes the number of solutions of instance A. Consider the following projection of $\mathsf{CIS^q}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. (This is the same projection as used for vertex cover.) Thus, we have

$$\mathsf{CIS^q}_n(z,t) = \sum_{T \subseteq E_n} z^{|T \cap E(A)|(q-1)} t^{(\text{\# vertices incident on } T)(q-1)}.$$

Now it follows easily that the number of cliques of size k, modulo p, is the coefficient of $z^{\binom{k}{2}(q-1)} t^{k(q-1)}$ in $\mathsf{CIS^q}_n(z,t)$.

(E′) Consider the weighted graph given by the values of $\tilde{X}$ and $\tilde{Y}$ variables. Each subset

$T \subseteq E_n$ contributes 0 or 1 to the sum. A subset $T \subseteq E_n$ contributes 1 to the sum if and only if all edges in T have non-zero weight, and every vertex incident on T must have non-zero weight. Therefore, we consider the graph induced on vertices with non-zero weights. Any subset of edges in this induced graph contributes 1 to the total sum; all other subsets contribute 0. Let ℓ be the number of edges in the induced graph with non-zero weights. Thus, the total sum is $2^\ell \bmod p$.

The *clow* polynomial $\mathsf{Clow}^{\mathsf{q}} = (\mathsf{Clow}^{\mathsf{q}}{}_n)$**:**

A clow in an n-vertex graph is a closed walk of length exactly n, in which the minimum numbered vertex (called the head) appears exactly once.

$$\mathsf{Clow}^{\mathsf{q}}{}_n := \sum_{w\text{: clow of length } n} \left(\prod_{e\text{: edges in } w} X_e^{q-1} \right) \left(\prod_{\substack{v\text{: vertices in } w \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

(If an edge e is used k times in a clow, it contributes $X_e^{k(q-1)}$ to the monomial.)

(H′) Given an instance $A = (V(A), E(A))$ of the Hamiltonian cycle problem with $|V(A)| = n$ and $|E(A)| = m$, we show how $\mathsf{Clow}^{\mathsf{q}}{}_n$ encodes the number of Hamiltonian cycles in A. Consider the following projection of $\mathsf{Clow}^{\mathsf{q}}{}_n$. Set $Y_v = t$, for $v \in V(A)$. For $e \in E(A)$, set $X_e = z$; otherwise $e \notin E(A)$ and set $X_e = 1$. (The same projection was used for VC^{q} and $\mathsf{CIS}^{\mathsf{q}}$.) Thus, we have

$$\mathsf{Clow}^{\mathsf{q}}{}_n(z,t) = \sum_{w\text{: clow of length } n} \left(\prod_{e\text{: edges in } w \cap E(A)} z^{q-1} \right) \left(\prod_{\substack{v\text{: vertices in } w \\ \text{(counted only once)}}} t^{q-1} \right).$$

From the definition, it now follows that number of Hamiltonian cycles in A, modulo p, is the coefficient of $z^{n(q-1)} t^{n(q-1)}$.

(E′) To evaluate $\mathsf{Clow}^{\mathsf{q}}{}_n$ on instantiations of $\tilde{X}$ and $\tilde{Y}$ variables, we consider the weighted graph given by the values to the variables. We modify the edge weights as follows: if an edge is incident on a node with zero weight, we make its weight 0 irrespective of the value of the corresponding X variable. Thus, all zero weight vertices are isolated in the modified graph G. Hence, the total sum is equal to the number of closed walks of length n, modulo p, in this modified graph. This can be computed in polynomial time using matrix powering as follows: Let G_i denote the induced subgraph of G with vertices $\{i, \ldots, n\}$, and let A_i be its adjacency matrix. We represent A_i as an $n \times n$ matrix with the first $i-1$ rows and columns having only zeroes. Now the number of clows with head i is given by the $[i, i]$ entry of $A_i A_{i+1}^{n-2} A_i$.

The *3D-matching* polynomial $\mathsf{3DM^q} = (\mathsf{3DM^q}_n)$:

Consider the complete tripartite hyper-graph, where each partition contain n nodes, and each hyperedge has exactly one node from each part. As before, there are variables X_e for hyperedge e and Y_v for node v.

$$\mathsf{3DM^q}_n := \sum_{M \subseteq A_n \times B_n \times C_n} \left(\prod_{e \in M} X_e^{q-1} \right) \left(\prod_{\substack{v \in M \\ \text{(counted only once)}}} Y_v^{q-1} \right).$$

(H′) Given an instance of 3D-Matching $\mathcal{H}$, we consider the usual projection. The variables corresponding to the vertices are all set to t. The edges present in $\mathcal{H}$ are all set to z, and the ones not present are set to 1. Then the number of 3D-matchings in $\mathcal{H}$, modulo p, is equal to the coefficient of $z^{n(q-1)} t^{3n(q-1)}$ in $\mathsf{3DM^q}_n(z, t)$.

(E′) To evaluate $\mathsf{3DM^q}_n$ over $\mathbb{F}_q$, consider the hypergraph obtained after removing the vertices with zero weight, edges with zero weight, and edges that contain a vertex with zero weight (even if the edges themselves have non-zero weight). Every subset of hyperedges in this modified hypergraph contributes 1 to the total sum, and all other subsets contribute 0. Hence, the evaluation equals $2^{(\text{\# edges in the modified hypergraph})} \bmod p$.

$\square$

Remark 4.3.1. *The above proof technique is specific to finite fields. Indeed the cut enumerator polynomial*

$$\sum_{S \subseteq [n]} \prod_{i \in S, j \in \overline{S}} x_{i,j}$$

where $x_{i,j} = x_{j,i}$, shown to be VNP*-intermediate over $\mathbb{F}_2$ [Bür99], is* VNP*-complete over the rationals $\mathbb{Q}$ [dRA12].*

4.4 Conclusion

For every finite field $\mathbb{F}_q$, we have shown a list of intermediate polynomials (Theorem 4.3.1) such that their definitions depend on the size q of the field. Motivated by the success of finding several natural intermediate families of polynomials, we believe the following open questions are of immediate importance :

- *Can we find families of polynomials, with integer coefficients, that are* VNP*-intermediate (under some natural complexity assumption of course) over all fields of characteristic p?*
- *Can we find families of polynomials, with integer coefficients, that are* VNP*-intermediate over all finite fields?, or fields with non-zero characteristic?*
- *Can we find an explicit family of polynomials, that is* VNP*-intermediate in characteristic zero?*
- *Is the family of polynomials $\mathsf{Clique}_n^{\log n}$* VNP*-intermediate, under some widely believed complexity assumption?*
- *Is there a family of polynomials that is "*VP*-intermediate"?* That is, it is in VP, but, under some plausible complexity assumption, neither in VBP nor VP-hard.

Part II

Boolean Function Analysis

Chapter 5

Boolean function analysis

5.1 Introduction

Fourier transforms are extensively used in a number of fields such as engineering, mathematics, and computer science. Within theoretical computer science, Fourier analysis of Boolean functions has evolved into one of the most useful and versatile tools. In particular, it has played an important role in establishing several results in complexity theory, learning theory, social choice, inapproximability, metric spaces, etc. See the book [O'D14] for a comprehensive survey of this area. (See de Wolf [dW08] for a short and nice introduction to Fourier analysis.)

Let $\widehat{f}$ denote the Fourier transform of a Boolean function $f\colon \{0,1\}^n \to \{+1,-1\}$. Then $\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1$ and hence we can define the (Shannon) entropy of the distribution given by $\widehat{f}(S)^2$:

$$\mathbb{H}(f) := \sum_{S\subseteq[n]} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2}. \tag{5.1}$$

Since entropy can not be more than the logarithm of the support size of the distribution, we have $0 \leqslant \mathbb{H}(f) \leqslant n$.

The notion of *influence* was studied by Ben-or and Linial [BL85] in the context of sharing an unbiased common random bit in the distributed setting. For a set $S \subseteq [n]$, the influence of S on f, $\mathsf{Inf}_S(f)$, is the probability that f is not constant upon setting all the variables *not* in S uniformly at random. In particular, when S is a singleton set, say $S = \{i\}$, then $\mathsf{Inf}_i(f)$ is the fraction of inputs at which the value of f gets flipped if we flip the i-th bit. The *total influence* of f, $\mathsf{Inf}(f)$, is defined as $\sum_{S\,:\,|S|=1} \mathsf{Inf}_S(f)$. Hence, intuitively, the total influence may be viewed as the expected number of coordinates of a random input which, when flipped, will cause the value of f to be changed.

For example, the Parity function on n variables has total influence n. That is, the parity function is never constant even when all but one of the variables are set. In particular, every variable has maximum possible influence of 1. Fourier expansion of Parity function is $(-1)^{\sum_{i\in[n]} x_i}$. Thus, it is easily seen that $\mathbb{H}(\mathsf{Parity})$ equals 0. Consider a dictator function $f(x_1, \ldots, x_i, \ldots, x_n) = (-1)^{x_i}$. It follows that the influence of the i-th variable is 1, whereas the rest of the variables have 0 influence. Thus, exactly one variable has high influence. Again, from the Fourier expansion, it follows $\mathbb{H}(\mathsf{dictator}) = 0$. Another interesting example is the Majority function, where each variable has *low* influence $\Theta(1/\sqrt{n})$, and, therefore, the total influence is $\Theta(\sqrt{n})$. It can also be shown that $\mathbb{H}(\mathsf{Majority}) = \Theta(\sqrt{n})$ (see, for instance, Section 5.3 in [O'D14]).

The Fourier Entropy-Influence (FEI) Conjecture, made by Friedgut and Kalai [FK96] in 1996, states that for every Boolean function, its Fourier entropy is bounded above by its total influence.

Fourier Entropy-Influence Conjecture: There exists a universal constant C such that for all $f : \{0,1\}^n \to \{+1,-1\}$,

$$\mathbb{H}(f) \leqslant C \cdot \mathsf{Inf}(f). \tag{5.2}$$

The conjecture intuitively asserts that if the Fourier coefficients of a Boolean function are “smeared out,” then its influence must be large, i.e., at a typical input, the value of f changes in several different directions. The original motivation for the conjecture stems from a study of threshold phenomena in random graphs. The existence of sharp thresholds for various graph properties is one of the significant discoveries in the theory of random graphs [ER60]. Friedgut and Kalai [FK96] asked *how large can the threshold interval be for a monotone graph property?*

Consider $f : \{0,1\}^n \to \{0,1\}$ representing a monotone graph property. Define $A_f(p) := \Pr[f(X_1, X_2, \ldots, X_n) = 1]$, where each X_i is an independent random variable that is 1 with probability p and 0 with probability $1 - p$. Let $\delta > 0$ be a small number. By threshold interval we mean the length of the interval $[p, q]$ such that $A_f(p)$ is δ, but $A_f(q)$ is $1 - \delta$. Then, the length of the threshold interval is inversely proportional to the derivative of $A_f(p)$, and by Russo's formula [Rus81, Mar74], the derivative of $A_f(p)$ equals the total influence of f (under the product measure where each bit is 1 with probability p and 0 otherwise). Hence, the graph property has a small threshold interval around p, that is, sharp threshold, if and only if it has large influence. Therefore, Friedgut and Kalai [FK96] asked for generic conditions that would force the influence to be large. Motivated by the Fourier-analytic formulae of the entropy and influence, they conjectured that a spread-out Fourier spectrum, i.e. large Fourier entropy, might be one such condition.

The FEI conjecture also has numerous applications [Kal]. In particular, it implies that for any n-vertex monotone graph property, the influence is at least $c(\log n)^2$. In other words, following the discussion in preceding paragraph it implies that for a monotone graph property on n vertices any threshold interval is of length at most $c'(\log n)^{-2}$. The best known upper bound, by Bourgain and Kalai [BK97], is $C_\epsilon(\log n)^{-2+\epsilon}$, for any $\epsilon > 0$. That is, a lower bound of $\Omega((\log n)^{2-\epsilon})$ on the influence of any n-vertex monotone graph property.

It also implies the existence of *sparse* real polynomial that approximates a Boolean function in L_2 norm. That is, there exists a polynomial $p \colon \mathbb{R}^n \to \mathbb{R}$ with at most $2^{O(\mathsf{Inf}(f)/\epsilon)}$

monomials such that $\mathbb{E}[(f(x) - p(x))^2] \leqslant \epsilon$. It is worth noting that Friedgut's junta theorem [Fri98] implies the existence of such sparse L_2-approximators, but with a weaker bound $2^{O(\mathsf{Inf}(f)^2/\epsilon^2)}$.

It further implies a variant of *Mansour's Conjecture* [Man95] stating that for a Boolean function computable by a DNF formula with m terms, most of its Fourier mass is concentrated on $\mathsf{poly}(m)$-many coefficients. A proof of Mansour's conjecture would imply a polynomial time *agnostic* learning algorithm for DNF's [GKK08] answering a major open question in computational learning theory.

In this chapter, we study the Fourier-Entropy Influence (FEI) conjecture, and prove various upper bounds on Fourier entropy of Boolean functions as well as general real-valued functions.

We give the basic definitions in Section 5.2. We then discuss upper bounds on entropy in terms of complexity measures larger than Influence in Section 5.3. Next in section 5.4 we establish a specific bound on Fourier entropy of polynomial threshold functions. We further prove the FEI conjecture for Read-Once formulas in Section 5.5. In section 5.6 we study entropy of real-valued functions.

5.2 Preliminaries

The objects of our study are functions defined on the Boolean hypercube $\{0, 1\}^n$. They might be Boolean-valued, that is, $f\colon \{0, 1\}^n \to \{+1, -1\}$, or real-valued $f\colon \{0, 1\}^n \to \mathbb{R}$. For most of the part, we will be concerned with Boolean-valued functions, and we will simply call them *Boolean functions*. We now recall some basic facts from query complexity and Fourier analysis. For a detailed treatment on query complexity please refer to [BdW02], while for Fourier analysis see [dW08, O'D14].

The set of all real functions on $\{0, 1\}^n$ is a 2^n-dimensional real vector space with an inner product defined by $\langle f, g\rangle = 2^{-n} \sum_{x\in\{0,1\}^n} f(x)g(x) = \mathbb{E}[f(x)g(x)]$, where the expectation is taken uniformly over all $x \in \{0,1\}^n$. The character functions $\chi_S(x) := (-1)^{\sum_{i\in S} x_i}$ for $S \subseteq [n]$ form an orthonormal basis for this space of functions with respect to the above inner product. Thus, every function $f\colon \{0,1\}^n \to \mathbb{R}$ has the *unique Fourier* expansion: $f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S(x)$. The vector $\widehat{f} = (\widehat{f}(S))_{S\subseteq[n]}$ is called the Fourier transform of the function f. The Fourier coefficient $\widehat{f}(S)$ of f at S is then given by $\widehat{f}(S) = \mathbb{E}[f(x)\chi_S(x)]$. The *degree* $\deg(f)$ of f is $\max\{|S| \mid \widehat{f}(S) \neq 0\}$. The norm of a function f is defined to be $\|f\| = \sqrt{\langle f, f\rangle}$. Then orthonormality of $\{\chi_S\}$ implies *Parseval's identity*: $\|f\|^2 = \sum_S \widehat{f}(S)^2$. In particular, for a Boolean function $f\colon \{0,1\}^n \to \{+1,-1\}$, we have $\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1$. Then the Fourier entropy $\mathbb{H}(f)$ of f is given by Equation 5.1. The *spectral norm* (or, L_1-norm), denoted $L_1(f)$, is given by $\sum_S |\widehat{f}(S)|$.

We recall that the *influence of f in the i-th direction*, denoted $\mathsf{Inf}_i(f)$, equals

$$\frac{|\{x \in \{0,1\}^n \colon f(x) \neq f(x \oplus e_i)\}|}{2^n},$$

where $x \oplus e_i$ is obtained from x by flipping the i-th bit of x. It is known that $\mathsf{Inf}_i(f) = \sum_{S\ni i} \hat{f}(S)^2$ [KKL88]. Thus, we have a formula for the *influence* of f: $\mathsf{Inf}(f) = \sum_{i=1}^n \mathsf{Inf}_i(f) = \sum_{S\subseteq[n]} |S|\hat{f}(S)^2$.

For $x \in \{0,1\}^n$, the *sensitivity* $\mathsf{s}_f(x)$ of f at x is $\#\{i \in [n] : f(x) \neq f(x \oplus e_i)\}$, i.e., the number of coordinates of x, which when flipped, will flip the value of f. The (maximum) *sensitivity* $\mathsf{s}(f)$ of the function f is the largest sensitivity of f at x over all $x \in \{0,1\}^n$, that is, $\mathsf{s}(f) := \max\{\mathsf{s}_f(x) : x \in \{0,1\}^n\}$. The *average sensitivity* $\mathsf{as}(f)$ of f is defined to be $2^{-n}\sum_{x\in\{0,1\}^n} s_f(x)$. It is easy to see that $\mathsf{Inf}(f) = \mathsf{as}(f)$ and hence we also have $\mathsf{as}(f) = \sum_{S\subseteq[n]} |S|\hat{f}(S)^2$.

The *block sensitivity* $\mathsf{bs}_f(x)$ of f on an input x is the maximum number of disjoint subsets $B_1, \ldots, B_t$ of $[n]$ such that for all $j \in [t]$, $f(x) \neq f(x \oplus e_{B_j})$, where e_{B_j} is the characteristic vector of the set B_j. The *block sensitivity* $\mathsf{bs}(f)$ is $\max_x \mathsf{bs}_f(x)$.

The *certificate complexity* $\mathsf{C}(f)$ measures how many of the variables have to be given a value in order to fix the value of f. More precisely, an f-certificate of an input x is a subset S of $[n]$ with an assignment $\alpha \in \{0,1\}^{|S|}$ such that $x|_S = \alpha$, and for all input y such that $y|_S = x|_S$, $f(x) = f(y)$. The *size* of a certificate is the cardinality of the subset S. The *certificate complexity* $\mathsf{C}_f(x)$ on an input x is the size of a smallest f-certificate for x. The *certificate complexity* $\mathsf{C}(f)$ of a function is $\max_x \mathsf{C}_f(x)$, and the *average certificate complexity* of f is defined to be $2^{-n} \sum_{x\in\{0,1\}^n} \mathsf{C}_f(x)$.

For an $\epsilon \in [0,1]$, the *noise sensitivity* $\mathsf{NS}_\epsilon(f)$ of f at ϵ is given by $\Pr_{x,y\sim_\epsilon x}[f(x) \neq f(y)]$, where x is chosen uniformly at random, and $y \sim_\epsilon x$ denotes that y is obtained by flipping each bit of x independently with probability ϵ. From the relationship between Fourier coefficients and noise sensitivity (see, for instance, [BKS99]), it follows that

$$\mathsf{NS}_\epsilon(f) = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]} (1-2\epsilon)^{|S|}\widehat{f}(S)^2.$$

Thus the derivative of $\mathsf{NS}_\epsilon(f)$ with respect to ϵ equals $\sum_{S\neq\emptyset} |S|(1-2\epsilon)^{|S|-1}\hat{f}(S)^2$. We shall denote the derivative by $\mathsf{NS}'_\epsilon(f)$.

A *decision tree* for a Boolean function f is a rooted binary tree in which each internal node is labeled with a variable x_i, and has two outgoing edges, labeled 0 and 1. Furthermore, each leaf is labeled with $+1$ or -1. On an input $x \in \{0,1\}^n$, the algorithm queries the tree in the following way to compute $f(x)$. It starts at the root. The root is labelled with some variable x_i. Based on the value of x_i in x, it either follows the 0-edge or the 1-edge. The algorithm proceeds recursively querying the subtree rooted at 0-edge or 1-edge, until it reaches a leaf. The output of the algorithm (or, tree) on x is the label of the leaf that is reached. We say that a decision tree computes f iff its output equals $f(x)$ for all x. The *complexity* of a decision tree is its depth, i.e., the number of queries made on the worst-case input. The *decision tree complexity* of f, denoted $D(f)$, is the depth of a minimal-depth decision tree that computes f. The *average depth* of a decision tree is the expected number of queries on a uniformly chosen random input, i.e., average length of a root to leaf path under the uniform distribution on inputs. Let $\bar{\mathsf{d}}(f)$ denote the minimum average depth of a decision tree computing f. The *size* of a decision tree is the number of leaves in it. The *leaf complexity* of f, denoted $L(f)$, is the size of a minimal-sized decision tree that computes f.

In more generalised decision trees, each node is allowed to query some (possibly complicated) function of some input bits. A particular case where each node is labeled by the parity of a subset of variables is called *parity* decision tree. Various complexity measures associated with decision trees can be generalised analogously to parity decision trees. In particular, the concept of average depth generalises as is, and hence we denote the minimum average depth of a parity decision tree computing f by $\oplus$-$\bar{\mathsf{d}}(f)$.

A *subcube* C of the cube $\{0,1\}^n$ is given by a mapping (partial assignment) $\alpha : [n] \to \{0,1,*\}$ and is defined to be the set of all vectors in $\{0,1\}^n$ that agree with α on coordinates *fixed*, i.e., assigned a non-$*$ value, by α. In other words, $C := C_\alpha := \{x \in \{0,1\}^n \colon \forall i \in [n], \alpha(i) \neq * \implies x_i = \alpha(i)\}$. We use $A := \{i \in [n] : \alpha(i) \neq *\}$ to denote the set of fixed coordinates of α and denote the cube C also by the pair (A, α). The cardinality of the set A is called the *co-dimension* of C, since $|C| = 2^{n-|A|}$.

For a function $f : \{0,1\}^n \to \{+1,-1\}$, a partition $C = \{C_1, \ldots, C_m\}$ of $\{0,1\}^n$ into subcubes C_i such that f is constant on each C_i is called a *(monochromatic) subcube partition* with respect to f. If C is a subcube partition monochromatic with respect to f, we also say C *computes* f. The number of subcubes in a partition C is called its *size*. We define the *co-dimension* of a subcube partition C as, $\max_i$ co-dimension(C_i). We denote by $L_c(f)$ the minimum number of subcubes in a subcube partition that computes f. Let us consider the following probability distribution over C where each C_i is chosen with probability $|C_i|/2^n$. If A_i denotes the set of coordinates fixed by C_i, then the probability mass associated with each C_i equals $1/2^{|A_i|}$. We define the *subcube-partition entropy* of C to be the (Shannon) entropy of the aforementioned distribution, that is, it equals $\sum_{i=1}^{m} \frac{|A_i|}{2^{|A_i|}}$.

We call a function f on n variables *non-degenerate* if it depends on all its variables, i.e.,

$\mathsf{Inf}_i(f) \neq 0, \forall i \in [n]$. It can be shown that any subcube partition C computing a non-degenerate function f on n variables must have size at least $n+1$. We will need the following theorem from [LLTY15].

Theorem 5.2.1 ([LLTY15])**.** *Suppose $f : \{0,1\}^n \to \{+1,-1\}$ is non-degenerate. Then there must exists an index $i \in [n]$ such that at least one of the restrictions $f_{\restriction x_i=0}$ or $f_{\restriction x_i=1}$ must be non-degenerate, i.e., depend on all the remaining variables in $[n] \setminus \{i\}$.*

Lemma 5.2.2. *Suppose $f : \{0,1\}^n \to \{+1,-1\}$ is non-degenerate. Then any subcube partition that computes f must have size at least $n+1$.*

Proof. We can now prove the lemma by induction on n. For $n = 1$, the claim is trivial since if the function depends on a variable, the variable and its complement must be in different (single point) subcubes. For $n > 1$, we note that since the function is non-degenerate, for *every* variable x_j, there must be at least one subcube fixing $x_j = 0$ and at least one subcube fixing $x_j = 1$. Now, let x_i be a variable given by the Theorem 5.2.1 such that, say, $f_{\restriction x_i=0}$ depends on all its $n-1$ variables. By induction, we must have at least n subcubes in the restricted partition computing $f_{\restriction x_i=0}$, where the restricted partition is obtained by restricting each of the subcubes in the original partition computing f to $x_i = 0$ half-cube. In the $x_i = 1$ half-cube, we must have at least one subcube, namely the one that restricts $x_i = 1$ in the original partition. All the n subcubes previously counted are disjoint from this since they either restricted $x_i = 0$ in the original partition or they didn't restrict x_i at all. So, all together we must have $n+1$ subcubes in the original partition computing f. $\square$

We say that a Boolean function $f\colon \{0,1\}^n \to \{+1,-1\}$ is a degree-d *threshold* function if there exists a degree-d (multilinear) polynomial $p(x_1,\ldots,x_n)$ over $\mathbb{R}$ such that $f(x) = sgn(p(x))$ for all $x \in \{0,1\}^n$, where $sgn(\theta) = +1$ if $\theta > 0$, and -1 if $\theta \leqslant 0$. Furthermore, there exists no degree $d-1$ polynomial that sign represents f.

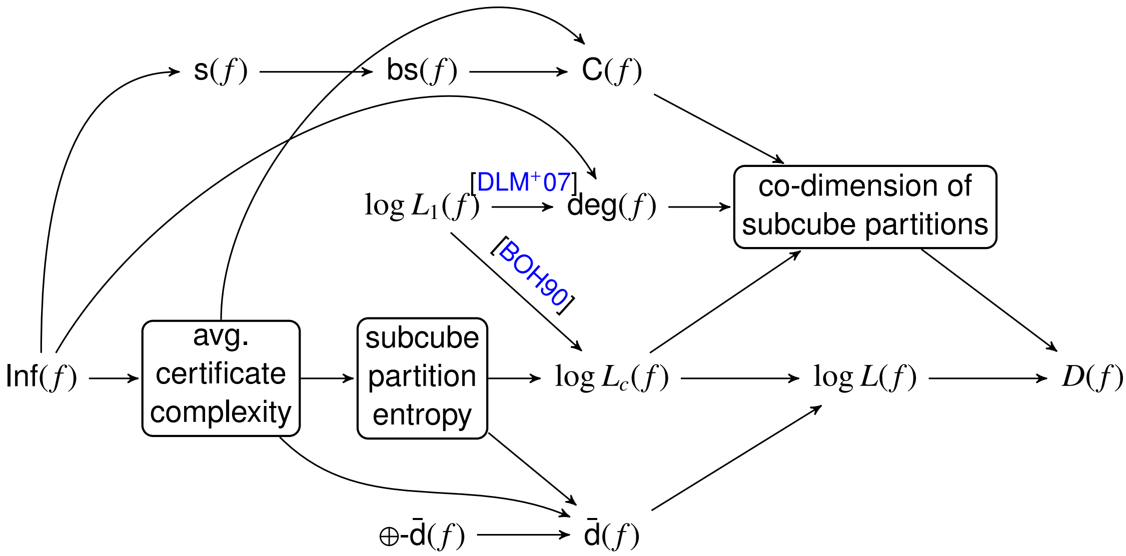

Figure 5.1: Relationship among complexity measures.

5.3 Upper bounds via Complexity measures

The $\mathsf{Inf}(f)$ of a Boolean function f is used to derive lower bounds on a number of complexity parameters of f such as the number of leaves, or the average depth of a decision tree computing f. For a detailed relationship among some complexity measures, see Fig. 5.1. An arrow from $A \to B$ implies $A = O(B)$. The unlabeled arrows follows from the definitions. The definitions of these measures are given in Section 5.2. Hence a natural weakening of the FEI conjecture is to prove upper bounds on the Fourier entropy in terms of such complexity measures.

In this section, we prove upper bounds on Fourier entropy in terms of some complexity parameters associated to decision trees and subcube partitions.

We begin with an easy-to-observe lemma; thus it can be considered folklore.

Lemma 5.3.1 (folklore)**.** *Let* $f\colon \{0,1\}^n \to \mathbb{R}$ *be such that* $\sum_S \widehat{f}(S)^2 = 1$*. Then,* $\mathbb{H}(f) = O(\log L_1(f))$*, where* $L_1(f)$ *is the* L_1*-norm of* f*.*

Proof. Let $L := L_1(f) = \sum_S |\widehat{f}(S)|$. Since $\sum_S \widehat{f}(S)^2 = 1$, we have $L \geqslant 1$. We prove the lemma in two cases: $L = 1$ and $L > 1$.

(*i*) $L = 1$: Using the fact $0 \leqslant |\widehat{f}(S)| \leqslant 1$, it can be shown that there is only one non-zero $\widehat{f}(S)$ with absolute value 1. Therefore, $\mathbb{H}(f) = 0 = \log L$.

(*ii*) $L > 1$: Let us define $\theta := 1/(16L^2)$, and $\mathcal{G} := \{S : |\widehat{f}(S)| \geqslant \theta\}$. Note that for $x \geqslant 16$, $\log x \leqslant \sqrt{x}$. We thus have $\log \frac{1}{|\widehat{f}(S)|} \leqslant \frac{1}{\sqrt{|\widehat{f}(S)|}}$, for $S \notin \mathcal{G}$. Therefore,

$$\begin{aligned}
\mathbb{H}(f) = \sum_{S} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2} &= \sum_{S \in \mathcal{G}} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2} + 2 \sum_{S \notin \mathcal{G}} \widehat{f}(S)^2 \log \frac{1}{|\widehat{f}(S)|} \\
&\leqslant \sum_{S \in \mathcal{G}} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2} + 2 \sum_{S \notin \mathcal{G}} \widehat{f}(S)^2 \frac{1}{\sqrt{|\widehat{f}(S)|}} \\
&\leqslant \left(\log \frac{1}{\theta^2}\right)\left(\sum_{S \in \mathcal{G}} \widehat{f}(S)^2\right) + 2\left(\max_{S \notin \mathcal{G}} \sqrt{|\widehat{f}(S)|}\right)\left(\sum_{S \notin \mathcal{G}} |\widehat{f}(S)|\right) \\
&\leqslant \log(256L^4) + 2 \cdot \frac{1}{4L} \cdot L = 4 \log L + 8.5.
\end{aligned}$$

Thus the lemma follows. □

Using the above lemma, we easily verify many easy-to-prove combinatorial bounds on Fourier entropy. In particular, we immediately have

$$\mathbb{H}(f) = O(\log L(f)), \ \mathbb{H}(f) = O(D(f)), \ \text{ and } \ \mathbb{H}(f) = O(\deg(f)). \tag{5.3}$$

We note that while Lemma 5.3.1 holds for real-valued functions as well, the inequalities in Eq. (5.3) hold *only for Boolean-valued* functions. We will give examples in Section 5.7 to show that these bounds fail for non-Boolean functions.

We now proceed to prove the main theorem of this section, Fourier entropy is bounded by $\oplus$-$\bar{\mathsf{d}}(f)$ (cf. Fig. 5.1). We will first establish a bound of $\bar{\mathsf{d}}(f)$, and then build on it to improve the bound to $\oplus$-$\bar{\mathsf{d}}(f)$.

Let T be a decision tree computing $f : \{0,1\}^n \to \{+1,-1\}$ on variable set $X = \{x_1, \ldots, x_n\}$. If $A_1, \ldots, A_L$ are the sets (with repetitions) of variables queried along the root-to-leaf paths in the tree T, then recall the average depth (w.r.t. the uniform distribution on inputs) of T is given by $\bar{\mathsf{d}} := \sum_{i=1}^{L} |A_i| 2^{-|A_i|}$. Observe that the average depth of a decision tree is also a kind of entropy: if each leaf λ_i is chosen with the probability $p_i = 2^{-|A_i|}$ that a uniformly chosen random input reaches it, then the entropy of the distribution induced on the λ_i is $\mathbb{H}(\lambda_i) = -\sum_i p_i \log p_i = \sum_i |A_i| 2^{-|A_i|}$. Here, we will show that *the Fourier entropy of f is at most twice the leaf entropy of a decision tree computing f.*

Without loss of generality, let x_1 be the variable queried by the root node of T and let T_1 and T_2 be the subtrees reached by the branches $x_1 = 0$ and $x_1 = 1$ respectively and let g_1 and g_2 be the corresponding functions computed on variable set $Y = X \setminus \{x_1\}$. Let $\bar{\mathsf{d}}$ be the average depth of T and $\bar{\mathsf{d}}_1$ and $\bar{\mathsf{d}}_2$ be the average depths of T_1 and T_2 respectively. We first observe a fairly straightforward lemma relating Fourier coefficients of f to the Fourier coefficients of restrictions of f.

Lemma 5.3.2. *Let $S \subseteq \{2, \ldots, n\}$.*

(i) $\widehat{f}(S) = (\widehat{g_1}(S) + \widehat{g_2}(S))/2$.

(ii) $\widehat{f}(S \cup \{1\}) = (\widehat{g_1}(S) - \widehat{g_2}(S))/2$.

(iii) $\bar{d} = (\bar{d}_1 + \bar{d}_2)/2 + 1$.

Proof. We observe the Fourier transform of f in terms of g_1 and g_2. It easily follows,

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) = f(x_1, y) &= \frac{(1 + (-1)^{x_1})}{2} g_1(y) + \frac{(1 - (-1)^{x_1})}{2} g_2(y) \\ &= \frac{g_1(y) + g_2(y)}{2} + (-1)^{x_1} \frac{g_1(y) - g_2(y)}{2}. \end{aligned}$$

(i) and (ii) now follow by linearity of the Fourier transform.

To establish (iii), we observe that while traversing the tree T on a uniformly random input, we traverse the left subtree or the right subtree with equal probabilities. It thus follows that $\bar{d} = 1 + (\bar{d}_1 + \bar{d}_2)/2$.

$\square$

Remark 5.3.1. *Note that g_1 and g_2 differ on an input y if and only if f is sensitive to x_1 at (x_1, y). In particular, it is easy to see $\frac{1}{4}\|g_1 - g_2\|^2 = \mathsf{Inf}_1(f)$ and $\frac{1}{4}\|g_1 + g_2\|^2 = 1 - \mathsf{Inf}_1(f)$.*

Remark 5.3.2. *We further remark that the proof of Lemma 5.3.2* (iii) *easily extends to the setting of parity decision trees. This will be of relevance later.*

We now recall a useful property of the function $-x \log x$.

Definition 5.3.3. *A function $h \colon \mathbb{R} \to \mathbb{R}$ is said to be* concave *over an interval* $[a, b]$ *if for every $p_1, p_2 \in [a, b]$ and $0 \leqslant \lambda \leqslant 1$, $h(\lambda p_1 + (1 - \lambda)p_2) \geqslant \lambda \cdot h(p_1) + (1 - \lambda) \cdot h(p_2)$.*

Fact 5.3.4. *The function $-x \log x$ is concave over* $[0, 1]$. *A particularly useful version is the following: for $x, y \in [0, 1]$,*

$$x \log \frac{1}{x} + y \log \frac{1}{y} \leqslant (x + y) \log \frac{2}{x + y}.$$

Using Lemma 5.3.2 and Fact 5.3.4 we establish the following technical lemma, which relates the entropy of f to entropies of restrictions of f.

Lemma 5.3.5. *Let g_1 and g_2 be defined as before in Lemma 5.3.2. Then,*

$$\mathbb{H}(f) \leqslant \frac{1}{2}\mathbb{H}(g_1) + \frac{1}{2}\mathbb{H}(g_2) + 2. \tag{5.4}$$

Proof. For simplicity of notation below, let $N' := \{2, \dots, n\}$.

$$\begin{aligned}
\mathbb{H}(f) &= \sum_{T \subseteq [n]} \widehat{f}(T)^2 \log \frac{1}{\widehat{f}(T)^2} \\
&= \sum_{S \subseteq N'} \left\{ \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2} + \widehat{f}(S \cup \{1\})^2 \log \frac{1}{\widehat{f}(S \cup \{1\})^2} \right\} \\
&\leqslant \sum_{S \subseteq N'} (\widehat{f}(S)^2 + \widehat{f}(S \cup \{1\})^2) \log \frac{2}{\widehat{f}(S)^2 + \widehat{f}(S \cup \{1\})^2} && \text{(by Fact 5.3.4)} \\
&= \sum_{S \subseteq N'} \frac{\widehat{g_1}(S)^2 + \widehat{g_2}(S)^2}{2} \log \frac{4}{\widehat{g_1}(S)^2 + \widehat{g_2}(S)^2} && \text{(by Lemma 5.3.2 (i) and (ii))}
\end{aligned}$$

$$= \frac{1}{2}\sum_{S\subseteq N'} \widehat{g_1}(S)^2 \log \frac{1}{\widehat{g_1}(S)^2 + \widehat{g_2}(S)^2} + \frac{1}{2}\sum_{S\subseteq N'} \widehat{g_2}(S)^2 \log \frac{1}{\widehat{g_1}(S)^2 + \widehat{g_2}(S)^2}$$

$$+ \sum_{S\subseteq N'} \left\{\widehat{g_1}(S)^2 + \widehat{g_2}(S)^2\right\}$$

$$\leqslant \frac{1}{2}\sum_{S\subseteq N'} \widehat{g_1}(S)^2 \log \frac{1}{\widehat{g_1}(S)^2} + \frac{1}{2}\sum_{S\subseteq N'} \widehat{g_2}(S)^2 \log \frac{1}{\widehat{g_2}(S)^2} + 2.$$

The last inequality follows from the monotonicity of Logarithm, and Parseval's identitiy, i.e., $\sum_{S\subseteq N'} \widehat{g_1}(S)^2 = \sum_{S\subseteq N'} \widehat{g_2}(S)^2 = 1$. □

Recall $\bar{\mathsf{d}}(f)$ denotes the minimum *average* depth of a decision tree computing f. As a consequence of Lemma 5.3.5 we obtain the following theorem.

Theorem 5.3.6. *For every Boolean function f, $\mathbb{H}(f) \leqslant 2 \cdot \bar{\mathsf{d}}(f)$.*

Proof. The proof is by induction on the number of variables of f.

Base case: $n = 1$. Then $\bar{\mathsf{d}}(f) = 0$, or 1. But in either case $\mathbb{H}(f) = 0$.

Induction Step:

$$\begin{aligned}
\mathbb{H}(f) &\leqslant \frac{1}{2}\mathbb{H}(g_1) + \frac{1}{2}\mathbb{H}(g_2) + 2 \qquad \text{(by Lemma 5.3.5)}\\
&\leqslant \bar{d}_1 + \bar{d}_2 + 2 \qquad \text{(by induction, } \mathbb{H}(g_i) \leqslant 2\bar{d}_i \text{ for } i = 1, 2)\\
&= 2\bar{d} \qquad \text{(by Lemma 5.3.2 (iii)).}
\end{aligned}$$

□

Further we observe that the constant 2 in the bound of Theorem 5.3.6 cannot be replaced by 1. Indeed, let $f(x, y) = x_1y_1 + \cdots + x_{n/2}y_{n/2} (\bmod 2)$ be the inner product mod 2 function. Then because $\widehat{f}(S)^2 = 2^{-n}$ for all $S \subseteq [n]$, $\mathbb{H}(f) = n$. On the other hand, it can be shown that $\bar{\mathsf{d}}(f) = \frac{3}{4}n - o(n)$. Hence, the constant must be at least $4/3$.

We now discuss the case of *parity* decision trees. The improved bound of parity decision trees (Theorem 5.3.8) and the discussion following it, also implies that the above proof technique cannot yield a constant factor better than 2 in Theorem 5.3.6.

For a linear transformation L and a Boolean function f, we define another Boolean function Lf as follows: $Lf(x) := f(Lx)$, for all $x \in \{0,1\}^n$. We begin with a useful observation.

Proposition 5.3.7 (folklore)**.** *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a Boolean function. For an invertible linear transformation $L \in \mathsf{GL}_n(\mathbb{F}_2)$, $\mathbb{H}(f) = \mathbb{H}(Lf)$.*

Proof. The proposition follows if we show that L permutes the Fourier-spectrum of f. Let us consider the Fourier coefficients of Lf. Let a row vector $y \in \{0,1\}^n$ denote a subset $S \subseteq [n]$, that is, $y_i = 1$ iff $i \in S$. Then,

$$\begin{aligned}\widehat{Lf}(y) &= \sum_{x\in\{0,1\}^n} Lf(x)\cdot(-1)^{\langle y,x\rangle} = \sum_{x\in\{0,1\}^n} f(Lx)\cdot(-1)^{\langle yL^{-1},Lx\rangle}\\ &= \sum_{z\in\{0,1\}^n} f(z)\cdot(-1)^{\langle yL^{-1},z\rangle} = \widehat{f}(yL^{-1}).\end{aligned}$$

□

Let T be a parity decision tree computing $f : \{0,1\}^n \to \{+1,-1\}$ on variable set $X = \{x_1,\ldots,x_n\}$. Also, let L be an invertible linear transformation. Note that a parity decision tree computing f also computes Lf and vice versa. This implies that, after applying a linear transformation, we can always assume that a variable is queried at the root node of T. Let us denote the new variable set, after applying the linear transformation, by $Y = \{y_1,\ldots,y_n\}$. Without loss of generality, let y_1 be the variable queried at the root. Let T_1 and T_2 be the subtrees reached by the branches $y_1 = 0$ and $y_1 = 1$ respectively, and let g_1 and g_2 be the corresponding functions computed on variable set $Y\setminus\{y_1\}$. Using Proposition 5.3.7, we see that the proofs of Lemma 5.3.2 (i), (ii), and Lemma 5.3.5 go through in the setting of parity decision trees too. We also remarked before that Lemma 5.3.2 (iii) holds. Hence, we get the following strengthening of Theorem 5.3.6.

Theorem 5.3.8. *For every Boolean function f, $\mathbb{H}(f) \leqslant 2\cdot\oplus\text{-}\bar{\mathsf{d}}(f)$.*

The constant 2 in the bound of Theorem 5.3.8 is optimal, that is, it cannot be replaced by a smaller number. As before, we consider the inner product mod 2 function. Its Fourier entropy is n, but $\left(\frac{n}{2}+1\right) \geqslant \oplus\text{-}\bar{\mathsf{d}}(f)$.

We now move on to discuss *subcube partitions* (see Section 5.2). The most natural subcube partitions with respect to a function f are the ones induced by decision trees computing f: the set of all inputs reaching a leaf of the decision tree is given by a subcube C_α, where α denotes the partial assignment defined by the path from the root to that leaf. But the subcube partition model allow any partitions, not only the one induced by a decision tree.

Suppose $f\colon \{0,1\}^n \to \{-1,0,1\}$ is computed by a subcube partition $C = \{C_1,\dots,C_L\}$, where $C_i = (A_i,\alpha_i)$. (C only needs to cover the non-zero inputs.) Let $\phi_i : \{0,1\}^n \to \{0,1\}$ be the characteristic function of the subcube C_i : $\phi_i(x) = 1$ if $x \in C_i$ and $\phi_i(x) = 0$ otherwise. Let $\beta_i \in \{-1,0,1\}$ be the value of f on C_i. Then, clearly

$$f(x) = \sum_{i=1}^{L} \beta_i \phi_i(x). \tag{5.5}$$

Using linearity of the Fourier transform, we also have the following relationship between Fourier coefficients and subcube partitions.

Proposition 5.3.9. *With f and C as defined above,* $\widehat{f}(S) = \sum_{i\,:\,S\subseteq A_i} 2^{-|A_i|}\cdot\beta_i\cdot\chi_S(\alpha_i)$.

Proof. From Eq. 5.5, using linearity, it follows that $\widehat{f}(S) = \sum_{i=1}^{L}\beta_i\widehat{\phi_i}(S)$.

Now a simple calculation shows that, for the characteristic function ϕ of a subcube $C = (A,\alpha)$, the Fourier transform is given by

$$\widehat{\phi}(S) = \begin{cases} 2^{-|A|}\chi_S(\alpha) & \text{if } S \subseteq A, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, it follows that $\widehat{f}(S) = \sum_{i\,:\,S\subseteq A_i} 2^{-|A_i|} \cdot \beta_i \chi_S(\alpha_i)$. In particular, $\widehat{f}(S) \neq 0 \implies \exists\, i\colon S \subseteq A_i$. □

The following lemma directly follows from the above discussions.

Lemma 5.3.10 ([BOH90])**.** *Let $f\colon \{0,1\}^n \to \{-1,0,1\}$ be computed by the subcube partition $C = \{C_1, \ldots, C_L\}$, where $C_i = (A_i, \alpha_i)$. Then,*

(i) $\sum_S |\widehat{f}(S)| \leqslant L$. *Hence, $L_1(f) \leqslant L_c(f)$.*

(ii) *For any integer $t \geqslant 0$, $\sum_{|S|\geqslant t} \widehat{f}(S)^2 \leqslant \sum_{|A_i|\geqslant t} 2^{-|A_i|}$.*

We reproduce a proof of (ii), since the proof technique will be of relevance in the proof of the next theorem, Theorem 5.3.11.

Proof. (of Lemma 5.3.10 (ii)): By Proposition 5.3.9, if $|S| \geqslant t$, the contribution to $\widehat{f}(S)$ comes from only the C_i such that $|A_i| \geqslant t$. Let $g \equiv \sum_{|A_i|\geqslant t} \beta_i \phi_i$ be the restriction of f to subcubes with co-dimension $\geqslant t$. It is then clear that

$$\sum_{|S|\geqslant t} \widehat{f}(S)^2 = \sum_{|S|\geqslant t} \widehat{g}(S)^2 \leqslant \sum_S \widehat{g}(S)^2 = 2^{-n} \sum_{|A_i|\geqslant t} |C_i| = \sum_{|A_i|\geqslant t} 2^{-|A_i|}.$$

The second equality follows from Parseval's identity. This proves (ii). □

Combining Lemma 5.3.10 (i) and Lemma 5.3.1 (see also Fig. 5.1), it immediately follows that $\mathbb{H}(f) = O(\log L_c(f))$. However, we give here a different approach to prove the same result. We believe this approach is more "natural" when compared to Lemma 5.3.1. It uses the concentration property of the Fourier transform and illustrates a general, potentially powerful, technique.

Theorem 5.3.11. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be computed by a subcube partition C of size L. Then,*

$$\mathbb{H}(f) \leqslant 2\log L + 2\log n + 2.$$

Proof. To bound entropy via concentration, we use the following simple idea. For a subset of coefficients $\mathcal{B}$, let $\mathbb{H}(\mathcal{B})$ denote the Fourier entropy restricted to that set $\mathcal{B}$, but appropriately normalized. That is,

$$\mathbb{H}(\mathcal{B}) = \sum_{S\in\mathcal{B}} \frac{\widehat{f}(S)^2}{\left(\sum_{T\in\mathcal{B}} \widehat{f}(S)^2\right)} \log \frac{\left(\sum_{T\in\mathcal{B}} \widehat{f}(S)^2\right)}{\widehat{f}(S)^2}.$$

Now if we suppose $\mathcal{E}$ is a subset of Fourier coefficients of a Boolean function f such that $\sum_{S\in\mathcal{E}} \widehat{f}(S)^2 = \epsilon$. Then a simple manipulation shows

$$\sum_S \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2} = (1-\epsilon)\,\mathbb{H}(\overline{\mathcal{E}}) + \epsilon\,\mathbb{H}(\mathcal{E}) + \mathrm{H}(\epsilon), \tag{5.6}$$

where $\mathrm{H}(p) := p\log\frac{1}{p} + (1-p)\log\frac{1}{1-p}$ is the binary entropy function.

Now, let $\mathcal{B}_t := \{S : \exists i |A_i| \leqslant t \text{ such that } S \subseteq A_i\}$. Note that if $S \notin \mathcal{B}_t$, then every set A_i that contains S must have size larger than t. Hence, using Proposition 5.3.9, only sets of size larger than t contribute to such $\widehat{f}(S)$. We now argue as in the proof of Lemma 5.3.10 (ii). Let $g \equiv \sum_{|A_i|>t} \beta_i\phi_i$ be the restriction of f to subcubes with co-dimension $> t$. It is then clear that

$$\sum_{S\notin\mathcal{B}_t} \widehat{f}(S)^2 = \sum_{S\notin\mathcal{B}_t} \widehat{g}(S)^2 \leqslant \sum_S \widehat{g}(S)^2 = 2^{-n} \sum_{|A_i|>t} |C_i| = \sum_{|A_i|>t} 2^{-|A_i|} < 2^{-t}L. \tag{5.7}$$

Since $\sum_i 2^{-|A_i|} = 1$, we have that $|\{i : |A_i| \leqslant t\}| \leqslant 2^t$. Since every $S \in \mathcal{B}_t$ is a subset of some A_i with $|A_i| \leqslant t$, it follows

$$|\mathcal{B}_t| \leqslant \sum_{|A_i|\leqslant t} 2^{|A_i|} \leqslant 2^t \cdot |\{i : |A_i| \leqslant t\}| \leqslant 2^{2t}. \tag{5.8}$$

Fix $t := \log(Ln)$. We can now estimate the Fourier entropy of a subcube partition:

$$\begin{aligned}
\mathbb{H}(f) &= \sum_S \hat{f}^2(S) \log \frac{1}{\hat{f}^2(S)} \\
&\leqslant (1 - 1/n)\, \mathbb{H}(\hat{f}^2(S) \ :\ S \in \mathcal{B}_t) + (1/n)\, \mathbb{H}(\hat{f}^2(S) \ :\ S \notin \mathcal{B}_t) + \mathrm{H}(1/n) \\
&\leqslant (1 - 1/n) \log |\mathcal{B}_t| + 1/n \cdot n + \mathrm{H}(1/n) \\
&\leqslant 2t + 1 + \mathrm{H}(1/n) \\
&\leqslant 2 \log L + 2 \log n + 2.
\end{aligned}$$

The second equality follows from using Eq. (5.6) and Eq. (5.7), and the next inequality follows from Eq. (5.8). □

Using Theorem 5.3.11 along with Lemma 5.2.2, we obtain the following corollary.

Corollary 5.3.12. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a Boolean function. Then, $\mathbb{H}(f) = O(\log L_c(f))$, where $L_c(f)$ is the minimum number of subcubes in a subcube partition that computes f.*

5.4 Polynomial Threshold functions

In this section, we establish a better upper bound on the Fourier entropy of polynomial threshold functions. We show that the Fourier entropy of a linear threshold function is $O(\sqrt{n})$, and we also show that for a degree-d threshold function it is $O_d(n^{1-\frac{1}{4d+6}})$.

It is well known [O'N71, AZ90, GL94] that the average sensitivity of a linear threshold function on n variables is $O(\sqrt{n})$. Moreover, majority over n bits Maj_n is a linear threshold function such that both $\mathsf{Inf}(\mathsf{Maj}_n)$ and $\mathbb{H}(\mathsf{Maj}_n)$ are $\Omega(\sqrt{n})$. Hence, solely as a function of n, the bound of the entropy cannot be improved. Also our upper bound on the Fourier entropy of degree-d threshold functions is of the same order of magnitude as the best known upper bound on their average sensitivity [HKM14, DRST14].

We note the facts discussed in the preceding paragraph to be used later.

Fact 5.4.1. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a Boolean function.*

(i) [O'N71, AZ90, GL94] *If f is a linear threshold function, $\mathsf{Inf}(f) \leqslant O(\sqrt{n})$.*

(ii) [HKM14] *If f is a degree-d threshold function, $\mathsf{Inf}(f) \leqslant 2^{O(d)} \cdot (n^{1-\frac{1}{4d+6}})$.* *(See also [DRST14].)*

For $f : \{0,1\}^n \to \{+1,-1\}$, let $W^k[f] := \sum_{|S|=k} \widehat{f}(S)^2$ and $W^{\geqslant k}[f] := \sum_{|S|\geqslant k} \widehat{f}(S)^2$. We first note a simple inequality, for a proof see, e.g., [O'D03], relating $W^{\geqslant 1/\epsilon}[f]$ and the noise sensitivity of f at ϵ (for definition, see Section 5.2).

Proposition 5.4.2. *For any $f : \{0,1\}^n \to \{+1,-1\}$, $\epsilon \in (0, \frac{1}{2}]$,*

$$W^{\geqslant(1/\epsilon)}[f] \equiv \sum_{S:|S|\geqslant 1/\epsilon} \widehat{f}(S)^2 \leqslant \frac{2}{1-e^{-2}} \mathsf{NS}_\epsilon(f),$$

where $\mathsf{NS}_\epsilon(f)$ is the noise sensitivity of f at ϵ.

Thus the above proposition suggests that upper bounds on noise sensitivity imply upper bounds on the tails of Fourier spectrum. Based on this intuition we prove our main technical lemma which translates a bound on noise sensitivity to a bound on the derivative (with respect to ϵ) of noise sensitivity.

Lemma 5.4.3. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be such that $\mathsf{NS}_\epsilon(f) \leqslant \alpha \cdot \epsilon^\beta$, where α is independent of ϵ and $\beta < 1$. Then,*

$$\mathsf{NS}'_\epsilon(f) \leqslant \frac{5}{1-e^{-2}} \cdot \frac{\alpha}{1-\beta} \cdot (1/\epsilon)^{1-\beta}.$$

Proof. We start with the formula for the derivative of noise sensitivity in terms of the Fourier weights.

$$\mathsf{NS}'_\epsilon(f) = \sum_{k=1}^{n} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1}$$

$$= \sum_{k=1}^{t} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1} + \sum_{k=t+1}^{n} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1}, \quad (t = \lfloor 1/\epsilon \rfloor)$$

$$\leqslant \sum_{k=1}^{t} W^k[f] \cdot k + \sum_{k=t}^{n} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1}. \tag{5.9}$$

Let $T_1 := \sum_{k=1}^{t} W^k[f] \cdot k$, and $T_2 := \sum_{k=t}^{n} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1}$. We will bound these sums individually using Proposition 5.4.2. We start with T_1.

$$\begin{aligned} T_1 = \sum_{k=1}^{t} W^k[f] \cdot k \leqslant \sum_{k=1}^{t} W^{\geqslant k}[f] &\leqslant \frac{2}{1-e^{-2}} \sum_{k=1}^{t} \mathsf{NS}_{\frac{1}{k}}(f) \\ &\leqslant \frac{2}{1-e^{-2}} \sum_{k=1}^{t} \alpha \cdot k^{-\beta} \simeq \frac{2}{1-e^{-2}} \cdot \alpha \cdot \frac{t^{1-\beta}}{1-\beta} \\ &\leqslant \frac{2}{1-e^{-2}} \cdot \frac{\alpha}{1-\beta} \cdot (1/\epsilon)^{1-\beta}. \end{aligned} \tag{5.10}$$

We now bound T_2.

$$\begin{aligned} T_2 &= \sum_{k=t}^{n} W^k[f] \cdot k \cdot (1-2\epsilon)^{k-1} \\ &\leqslant t \cdot W^{\geqslant t}[f] \cdot (1-2\epsilon)^{t-1} + \sum_{k \geqslant t+1} (1-2\epsilon)^{k-1} W^{\geqslant k}[f] \\ &\leqslant \frac{2}{1-e^{-2}} \left[t \cdot \mathsf{NS}_{\frac{1}{t}}(f) + \sum_{k \geqslant t+1} (1-2\epsilon)^{k-1} \mathsf{NS}_{\frac{1}{k}}(f) \right] \\ &\leqslant \frac{2}{1-e^{-2}} \left[t \cdot \alpha \cdot t^{-\beta} + \sum_{k \geqslant t+1} (1-2\epsilon)^{k-1} \cdot \alpha \cdot k^{-\beta} \right] \\ &\leqslant \frac{2}{1-e^{-2}} \left[\alpha \cdot t^{1-\beta} + \alpha \cdot (t+1)^{-\beta} \sum_{k \geqslant t+1} (1-2\epsilon)^{k-1} \right] \\ &\leqslant \frac{2}{1-e^{-2}} \left[\alpha \cdot t^{1-\beta} + \alpha \cdot (t+1)^{-\beta} \cdot \frac{(1-2\epsilon)^t}{2\epsilon} \right] \\ &\leqslant \frac{3}{1-e^{-2}} \cdot \alpha \cdot (1/\epsilon)^{1-\beta}. \end{aligned} \tag{5.11}$$

Using Eq. (5.10) and Eq. (5.11), in Eq. (5.9), we obtain the claimed bound in the lemma.

$\square$

From [OWZ11] we have the following bound on entropy.

Lemma 5.4.4. *Let* $f : \{0,1\}^n \to \{+1,-1\}$ *be a Boolean function. Then,*

$$\mathbb{H}(f) \leqslant (3 + \log_2 e) \cdot \mathsf{Inf}(f) + \log_2 e \cdot \sum_{k=1}^{n} W^k[f] k \ln \frac{n}{k}.$$

This lemma suggests that one way to prove a non-trivial upper bound on Fourier entropy is to bound the second summand on the right in a general way. Using Lemma 5.4.3, we prove another technical lemma that provides a bound on $\sum_{k=1}^{n} W^k[f] k \ln \frac{n}{k}$.

Lemma 5.4.5. *Let* $f : \{0,1\}^n \to \{+1,-1\}$ *be a Boolean function. Then,*

$$\sum_{k=1}^{n} W^k[f] k \ln \frac{n}{k} \leqslant exp(1/2) \cdot \frac{5}{1-e^{-2}} \cdot \frac{\alpha}{(1-\beta)^2} \cdot (4n)^{1-\beta},$$

where α *and* β *are as defined in Lemma 5.4.3.*

Proof. The first few steps of inequalities below are the same as in [OWZ11].

$$\begin{aligned}
\sum_{k=1}^{n} W^k[f] k \ln \frac{n}{k} &\leqslant \sum_{k=1}^{n} W^k[f] k \cdot \sum_{j=k}^{n} \frac{1}{j} \leqslant \sum_{j=1}^{n} \frac{1}{j} \sum_{k=1}^{j} W^k[f] k \\
&\leqslant \sum_{j=1}^{n} \frac{1}{j} \sum_{k=1}^{j} W^k[f] k \cdot \exp(1/2)(1-\frac{1}{2j})^{k-1}, \\
&\qquad \left[\text{since } \exp(1/2)(1-\tfrac{1}{2j})^m \geqslant 1, \forall m \leqslant (j-1)\right] \\
&\leqslant \sum_{j=1}^{n} \frac{1}{j} \cdot \exp(1/2) \cdot \mathsf{NS}'_{\frac{1}{4j}}(f) \\
&\leqslant exp(1/2) \cdot \frac{5}{1-e^{-2}} \cdot \frac{\alpha}{1-\beta} \cdot \sum_{j=1}^{n} \frac{1}{j} \cdot (4j)^{1-\beta} \\
&\leqslant exp(1/2) \cdot \frac{5}{1-e^{-2}} \cdot \frac{\alpha}{1-\beta} \cdot 4^{1-\beta} \cdot \sum_{j=1}^{n} j^{-\beta} \\
&\leqslant exp(1/2) \cdot \frac{5}{1-e^{-2}} \cdot \frac{\alpha}{(1-\beta)^2} \cdot 4^{1-\beta} \cdot n^{1-\beta}. \qquad (5.12)
\end{aligned}$$

□

Using Lemma 5.4.5 and Lemma 5.4.4, we obtain the following theorem which bounds the Fourier entropy of a Boolean function.

Theorem 5.4.6. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a Boolean function such that $\mathsf{NS}_\epsilon(f) \leqslant \alpha \cdot \epsilon^\beta$. Then*

$$\mathbb{H}(f) \leqslant C \cdot \left(\mathsf{Inf}(f) + \frac{\alpha}{(1-\beta)^2} \cdot (4n)^{1-\beta} \right) ,$$

where C is a universal constant.

In particular, for polynomial threshold functions, there exist non-trivial bounds on their noise sensitivity.

Theorem 5.4.7 (Peres's Theorem [O'D03])**.** *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a linear threshold function. Then $\mathsf{NS}_\epsilon(f) \leqslant O(\sqrt{\epsilon})$.*

Theorem 5.4.8 ([HKM14])**.** *For any degree-d polynomial threshold function $f : \{0,1\}^n \to \{+1,-1\}$ and $0 < \epsilon < 1$, $\mathsf{NS}_\epsilon(f) \leqslant 2^{O(d)} \cdot \epsilon^{1/(4d+6)}$.*

As corollaries of Theorem 5.4.6, using Fact 5.4.1 with Theorem 5.4.7 and Theorem 5.4.8, we obtain the following bounds on the Fourier entropy of polynomial threshold functions.

Corollary 5.4.9. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a linear threshold function. Then, $\mathbb{H}(f) \leqslant C \cdot \sqrt{n}$, where C is a universal constant.*

Proof. It follows from Theorem 5.4.7 that we can choose $\beta = 1/2$, and α a universal constant in Theorem 5.4.6. Now using Fact 5.4.1 (i) we obtain the corollary. □

Similarly, we establish the following bound for general polynomial threshold functions.

Corollary 5.4.10. *Let $f : \{0,1\}^n \to \{+1,-1\}$ be a degree-d polynomial threshold function. Then, $\mathbb{H}(f) \leqslant C \cdot 2^{O(d)} \cdot n^{1-\frac{1}{4d+6}}$, where C is a universal constant.*

5.5 Read-Once Formulas

In this section, we will prove the Fourier Entropy-Influence conjecture for read-once formulas using AND, OR, XOR, and NOT gates. We mention that our result is subsumed by a concurrent and independent work of O'Donnell and Tan [OT13]. Using a completely different technique, they proved the conjecture for read-once formulas with *arbitrary* gates of bounded fan-in.

It is well-known that both Fourier entropy and average sensitivity add up when two functions on disjoint sets of variables are added modulo 2.

Fact 5.5.1. *Let* $f = g_1 \oplus g_2$ *for* $g_i : \{0,1\}^{V_i} \to \{-1,+1\}$*, where* $V_1 \cap V_2 = \emptyset$*. Then,*

1. $\mathbb{H}(f) = \mathbb{H}(g_1) + \mathbb{H}(g_2)$
2. $\mathsf{as}(f) = \mathsf{as}(g_1) + \mathsf{as}(g_2)$.

We will show that somewhat analogous "tensorizability" properties hold when composing functions on disjoint sets of variables using AND and OR operations.

For $f : \{0,1\}^n \to \{+1,-1\}$, let $f_{\mathbb{B}}$ denote its 0-1 counterpart: $f_{\mathbb{B}} \equiv \frac{1-f}{2}$. Let us define the following 0-1 variant of $\mathbb{H}$:

$$\mathbf{H}(f_{\mathbb{B}}) := \sum_S \widehat{f_{\mathbb{B}}}(S)^2 \log \frac{1}{\widehat{f_{\mathbb{B}}}(S)^2}. \tag{5.13}$$

Note that $\mathbf{H}(f_{\mathbb{B}})$ is not exactly an entropy. An easy relation enables translation between $\mathbb{H}(f)$ and $\mathbf{H}(f_{\mathbb{B}})$:

Lemma 5.5.2. *Let* $p := \Pr[f_{\mathbb{B}} = 1] = \widehat{f_{\mathbb{B}}}(\emptyset) = \sum_S \widehat{f_{\mathbb{B}}}(S)^2$ *and* $q := 1 - p$*. Then,*

$$\mathbb{H}(f) = 4 \cdot \mathbf{H}(f_{\mathbb{B}}) + \varphi(p), \quad \textit{where} \tag{5.14}$$

$$\varphi(p) := \mathrm{H}(4pq) - 4p(\mathrm{H}(p) - \log p). \tag{5.15}$$

$\mathrm{H}(p)$ *is the binary entropy function which also equals* $p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$.

Now, let $f = \mathsf{AND}(g_1, g_2)$ for $g_i : \{0,1\}^{V_i} \to \{-1,+1\}$, where $V_1 \cap V_2 = \varnothing$. Let $V = V_1 \cup V_2$. Let $g_{i\mathbb{B}} \equiv \frac{1-g_i}{2}$ and $p_i = \widehat{g_{i\mathbb{B}}}(\varnothing)$. It is then obvious that $f_{\mathbb{B}} \equiv g_{1\mathbb{B}} \cdot g_{2\mathbb{B}}$.

Lemma 5.5.3. *With the above notations, the following identities hold:*

1. *For all* $S \subseteq V$, $\widehat{f_{\mathbb{B}}}(S) = \widehat{g_{1\mathbb{B}}}(S \cap V_1) \cdot \widehat{g_{2\mathbb{B}}}(S \cap V_2)$
2. $\mathbf{H}(f_{\mathbb{B}}) = p_2 \cdot \mathbf{H}(g_{1\mathbb{B}}) + p_1 \cdot \mathbf{H}(g_{2\mathbb{B}})$
3. $\mathsf{as}(f) = p_2 \cdot \mathsf{as}(g_1) + p_1 \cdot \mathsf{as}(g_2)$.

Proof. Proof of this lemma follows by direct computation and hence omitted. □

$$\text{For } 0 \leqslant p \leqslant 1, \text{ we define:} \quad \psi(p) := p^2 \log \frac{1}{p^2} - 2\,\mathrm{H}(p), \text{ and } q := 1 - p. \tag{5.16}$$

Before going on, we pause to give some intuition about the choice of the function ψ and the function κ below in Eq. (5.19). In the FEI conjecture (Eq. (5.2)), the right hand side, $\mathsf{Inf}(f)$, does not depend on whether we take the range of f to be $\{-1,+1\}$ or $\{0,1\}$. In contrast, the left hand side, $\mathbb{H}(f)$, depends on the range being $\{-1,+1\}$. Just as the usual entropy-influence inequality composes with respect to the parity operation (Fact 5.5.1) with $\{-1,+1\}$ range, we expect a corresponding composition with $\{0,1\}$ range to hold for the AND operation (and by symmetry for the OR operation). However, Lemma 5.5.2 shows the translation to $\{0,1\}$-valued functions results in the annoying additive "error" term $\varphi(p)$. Such additive terms that depend on p create technical difficulties in the inductive proofs below and we need to choose the appropriate functions of p carefully.

For example, we know $4\,\mathbf{H}(f_{\mathbb{B}}) + \varphi(p) = \mathbb{H}(f) = 4\,\mathbf{H}(1 - f_{\mathbb{B}}) + \varphi(q)$ from Lemma 5.5.2. If the conjectured inequality for the $\{0,1\}$-valued entropy-influence inequality has an additive error term $\psi(p)$ (see Eq. (5.17) below), then we must have $\mathbf{H}(f_{\mathbb{B}}) - \mathbf{H}(1 - f_{\mathbb{B}}) =$

$\psi(p) - \psi(q) = (\varphi(q) - \varphi(p))/4 = p^2 \log \frac{1}{p^2} - q^2 \log \frac{1}{q^2}$, using Eq. (5.15). Hence, we may conjecture that $\psi(p) = p^2 \log \frac{1}{p^2}$ + (an additive term symmetric with respect to p and q). Given this and the other required properties, e.g., Lemma 5.5.4 below, for the composition to go through, we are led to the definition of ψ in Eq. (5.16). Similar considerations with respect to composition by parity operation (in addition to those by AND, OR, and NOT) leads us to the definition of κ in Eq. (5.19).

Let us define the **FEI01 Inequality** (the 0-1 version of FEI) as follows:

$$\mathbf{H}(f_{\mathbb{B}}) \leqslant c \cdot \mathsf{as}(f) + \psi(p), \tag{5.17}$$

where $p = \widehat{f_{\mathbb{B}}}(\emptyset) = \Pr_x[f_{\mathbb{B}}(x) = 1]$ and c is a constant to be fixed later.

The following technical lemma gives us the crucial property of ψ:

Lemma 5.5.4. *For ψ as in Eq. (5.16) and $p_1, p_2 \in [0, 1]$, $p_1\psi(p_2) + p_2\psi(p_1) \leqslant \psi(p_1 p_2)$.*

Since the proof of the lemma is somewhat technical, we move the proof to the end of this section. Given this lemma, an inductive proof shows that the Fourier entropy-influence conjecture holds for read-once formulas over the complete basis of {AND, OR, NOT}. We now complete the steps of the inductive proof.

Lemma 5.5.5. *Suppose $f_{\mathbb{B}} = \mathsf{AND}(g_{1\mathbb{B}}, g_{2\mathbb{B}})$, where the g_i's depend on* disjoint *sets of variables. If each of the g_i satisfies the FEI01 Inequality (5.17), then so does f.*

Proof.

$$\begin{aligned}
\mathbf{H}(f_{\mathbb{B}}) &= p_2\, \mathbf{H}(g_{1\mathbb{B}}) + p_1\, \mathbf{H}(g_{2\mathbb{B}}) \quad \text{by Lemma 5.5.3 (2)}\\
&\leqslant p_2(c \cdot \mathsf{as}(g_1) + \psi(p_1)) + p_1(c \cdot \mathsf{as}(g_2) + \psi(p_2)) \quad \text{since } g_i \text{ satisfy Eq. (5.17)}\\
&= c \cdot (p_2\mathsf{as}(g_1) + p_1\mathsf{as}(g_2)) + (p_2\psi(p_1) + p_1\psi(p_2)\\
&\leqslant c \cdot \mathsf{as}(f) + \psi(p) \quad \text{by Lemma 5.5.3 (3) and Lemma 5.5.4}
\end{aligned}$$

□

Lemma 5.5.6. *If f satisfies FEI01 inequality (5.17), then so does its negation, i.e., $1 - f$.*

Proof. Note that $\mathbf{H}(1-f) = \mathbf{H}(f) - p^2 \log \frac{1}{p^2} + q^2 \log \frac{1}{q^2}$ and because $\mathrm{H}(p) = \mathrm{H}(q)$, $\psi(p) - \psi(q) = p^2 \log \frac{1}{p^2} - q^2 \log \frac{1}{q^2}$. □

Corollary 5.5.7. *Suppose $f_{\mathbb{B}} = \mathsf{OR}(g_{1\mathbb{B}}, g_{2\mathbb{B}})$, where the g_i depend on* disjoint *sets of variables. If each of the g_i satisfies the FEI01 Inequality (5.17), then so does f.*

Proof. Note that $1 - f_{\mathbb{B}} = (1 - g_{1\mathbb{B}}) \cdot (1 - g_{2\mathbb{B}})$ and apply lemmas, Lemma 5.5.5 and Lemma 5.5.6. □

Theorem 5.5.8. *The FEI01 inequality (5.17) holds for all read-once Boolean formulas using* AND, OR, *and* NOT *gates, with constant $c = 5/2$.*

Proof. Let f be computed by a read-once Boolean formula. We assume without loss of generality that negations only appear at the bottom with leaves. We proceed by induction on the underlying tree. At the leaves f is a literal associated with a single variable, say x_1. Then, since $f_{\mathbb{B}}(\varnothing) = 1/2$ and $f_{\mathbb{B}}(\{1\}) = -1/2$, we calculate $\mathbf{H}(f_{\mathbb{B}}) = \frac{1}{4} \log 4 + \frac{1}{4} \log 4 = 1$, $\mathsf{as}(f) = 1$, $p = 1/2$, and $\psi(1/2) = -3/2$. Thus with $c = 5/2$, Eq. (5.17) is satisfied.

Now, Lemma 5.5.5 and Corollary 5.5.7 imply that at every AND gate and OR gate, the inequality (5.17) is preserved, i.e., if it holds at both the inputs, it also holds at the output. □

We now proceed to show that the above result can be extended to read-once formulas that include XOR gates as well. To switch to the usual FEI inequality (in the $\{-1, +1\}$ notation), we combine Eq. (5.17) and Eq. (5.14) to obtain

$$\mathbb{H}(f) \leqslant 10 \cdot \mathsf{as}(f) + \kappa(p), \quad \text{where} \tag{5.18}$$

$$\kappa(p) := 4\psi(p) + \varphi(p) = -8\,\mathrm{H}(p) - 8pq - (1 - 4pq)\log(1 - 4pq). \tag{5.19}$$

Since it uses the $\{-1,+1\}$ range, we expect that Eq. (5.18) should be preserved by parity composition of functions. The only technical detail is to show that the function κ also behaves well with respect to parity composition. We show that this indeed happens. Consider $f \equiv g_1 \oplus g_2$. Since parity is a simple product over $\{-1,+1\}$ range we have $f = g_1 \cdot g_2$, and therefore, $p = p_1 q_2 + p_2 q_1$. Thus we only need to show the following lemma.

Lemma 5.5.9. *For κ as defined by Eq. (5.19), $\kappa(p_1) + \kappa(p_2) \leqslant \kappa(p_1 q_2 + p_2 q_1)$.*

Again, due to the technical nature of the proof, we move it to the end of the section. We can now prove the following composition lemma which leads us to the main theorem of this section.

Lemma 5.5.10. *Suppose $f = g_1 \cdot g_2$, where the g_i depend on* disjoint *sets of variables. If each of the g_i satisfies the entropy-influence inequality (5.18), then so does f.*

Proof.

$$
\begin{aligned}
\mathbb{H}(f) &= \mathbb{H}(g_1) + \mathbb{H}(g_2) && \text{by Fact 5.5.1 (i)} \\
&\leqslant 10 \cdot \mathsf{as}(g_1) + \kappa(p_1) + 10 \cdot \mathsf{as}(g_2) + \kappa(g_2) && \text{since } g_i \text{ satisfy Eq. (5.18)} \\
&= 10 \cdot \mathsf{as}(f) + \kappa(p_1) + \kappa(p_2) && \text{by Fact 5.5.1 (ii)} \\
&\leqslant 10 \cdot \mathsf{as}(f) + \kappa(p) && \text{by Lemma 5.5.9.}
\end{aligned}
$$

□

Theorem 5.5.11. *If f is computed by a read-once formula using* AND*,* OR*,* XOR*, and* NOT *gates, then $\mathbb{H}(f) \leqslant 10\,\mathsf{Inf}(f) + \kappa(p)$.*

Proof. We use induction on the tree given by the formula computing f to prove Eq. (5.18). Without loss of generality we assume that negations are only at the bottom with leaves. So the leaves are input variables or their negations and the claim that they satisfy Eq. (5.18) can be verified by direct calculation. At any internal node, its two inputs are given by subformulas depending on disjoint sets of variables by the read-once property of the formula. When the internal node is an AND or OR gate, the claim follows from Eq. (5.14), Lemma 5.5.5, Corollary 5.5.7, and Eq. (5.19). When the internal node is an XOR gate, the claim follows from Lemma 5.5.10. Thus Eq. (5.18) holds at the root of the tree and hence for f. □

Observe that the parity function on n variables shows that the bound in Theorem 5.5.11 is tight. But, it is not tight without the additive term $\kappa(p)$. Further it is easy to verify that $-10 \leqslant \kappa(p) \leqslant 0$ for $p \in [0,1]$. Hence the theorem implies $\mathbb{H}(f) \leqslant 10\,\mathsf{Inf}(f)$ for all read-once formulas f using AND, OR, XOR, and NOT gates.

We now give the proofs of the two technical lemmas, Lemma 5.5.4 and Lemma 5.5.9.

Lemma 5.5.4 restated: For ψ as defined by Eq. (5.16) and $p_1, p_2 \in [0,1]$,

$$p_1 \cdot \psi(p_2) + p_2 \cdot \psi(p_1) \leqslant \psi(p_1 p_2).$$

Proof. We need to prove that $p_1\psi(p_2) + p_2\psi(p_1) - \psi(p_1 p_2) \leqslant 0$. Let us define $q_1 := 1 - p_1$ and $q_2 := 1 - p_2$. We begin by manipulating the left hand side:

$$\begin{aligned}
&p_1\psi(p_2) + p_2\psi(p_1) - \psi(p_1p_2)\\
&= p_1\left(p_2^2 \log\frac{1}{p_2^2} - 2\,\mathrm{H}(p_2)\right) + p_2\left(p_1^2\log\frac{1}{p_1^2} - 2\,\mathrm{H}(p_1)\right) - (p_1p_2)^2\log\frac{1}{(p_1p_2)^2} + 2\,\mathrm{H}(p_1p_2)\\
&= 2p_1p_2\left(-p_2\log p_2 - p_1\log p_1 + p_1p_2\log p_2 + p_1p_2\log p_1\right) + 2\left(\mathrm{H}(p_1p_2) - p_2\,\mathrm{H}(p_1) - p_1\,\mathrm{H}(p_2)\right)\\
&= 2p_1p_2\left(-p_2q_1\log p_2 - p_1q_2\log p_1\right) + 2\left(-(1-p_1p_2)\log(1-p_1p_2) + p_2q_1\log q_1 + p_1q_2\log q_2\right)\\
&= 2p_1q_2\left(-p_1p_2\log p_1 + \log q_2\right) + 2p_2q_1\left(-p_1p_2\log p_2 + \log q_1\right) - 2(1-p_1p_2)\log(1-p_1p_2)\\
&\leqslant 2(1-p_1p_2)\left(-p_1p_2\log(p_1p_2) + \log\frac{q_1q_2}{1-p_1p_2}\right) \quad \text{since } p_1q_2, p_2q_1 \leqslant (1-p_1p_2)\\
&\leqslant 2(1-p_1p_2)\left(-p_1p_2\log(p_1p_2) + \log\frac{(1-\sqrt{p_1p_2})^2}{(1-p_1p_2)}\right)\\
&\qquad \text{since } q_1q_2 = (1-p_1)(1-p_2) \leqslant (1-\sqrt{p_1p_2})^2,\\
&\qquad \text{e.g., by the AM-GM inequality } p_1 + p_2 \geqslant 2\sqrt{p_1p_2}.
\end{aligned}$$

Since $p_1 p_2 \in [0,1]$, it suffices to show the (univariate) inequality $\tau(x) := -x \ln x + \ln \frac{(1-\sqrt{x})^2}{1-x} \leqslant 0$ for $x \in [0,1]$. Since the boundary cases are easy to verify, it suffices to prove the that $\tau(x) \leqslant 0$ for $x \in (0,1)$. Note that $\tau(0) = 0$ and hence it suffices to prove that $\tau'(x) < 0$ for $x \in (0,1)$. But

$$\begin{aligned}
\tau'(x) &= -1 + \ln \frac{1}{x} - \frac{1}{\sqrt{x}(1-x)} \\
&\leqslant -1 + \sqrt{\frac{1}{x}} - \frac{1}{\sqrt{x}(1-x)} \quad \text{since } \ln y \leqslant \sqrt{y} \\
&= -1 - \frac{\sqrt{x}}{1-x} \\
&< 0 \quad \text{for } x \in (0,1).
\end{aligned}$$

□

Lemma 5.5.9 restated: For κ as defined by Eq. (5.19) and $p_1, p_2 \in [0,1]$,

$$\kappa(p_1) + \kappa(p_2) \leqslant \kappa(p_1 q_2 + p_2 q_1),$$

where $q_1 = 1 - p_1$, and $q_2 = 1 - p_2$.

Proof. In the following, we will let $p = p_1 q_2 + p_2 q_1$, and $q = 1 - p = p_1 p_2 + q_1 q_2$.

To begin with, we observe that $(1 - 4pq) = (p-q)^2$ and that $(p - q) = (p_1 - q_1)(p_2 - q_2)$, i.e., parity operation on independent Boolean variables results in multiplying their *biases*, and hence $(1 - 4pq) = (1 - 4p_1 q_1)(1 - 4p_2 q_2)$. Using this, we relate the third terms on either side of the inequality to be proved.

$$\begin{aligned}
(1 - 4pq)\log(1 - 4pq) &= (1 - 4p_1 q_1)(1 - 4p_2 q_2)\log((1 - 4p_1 q_1)(1 - 4p_2 q_2)) \\
&= (1 - 4p_2 q_2)\,((1 - 4p_1 q_1)\log(1 - 4p_1 q_1)) \\
&\quad + (1 - 4p_1 q_1)\,((1 - 4p_2 q_2)\log(1 - 4p_2 q_2)) \\
&\leqslant (1 - 4p_1 q_1)\log(1 - 4p_1 q_1) + (1 - 4p_2 q_2)\log(1 - 4p_2 q_2)
\end{aligned}$$

$$+ 64p_1q_1p_2q_2,$$

The last inequality follows from the fact $-(1-4p_iq_i)\log(1-4p_iq_i) \leqslant 8p_iq_i$, which in turn follows from the inequality $x\log\frac{1}{x} \leqslant$ [1] $2(1-x)$ for $x \in [0,1]$. Thus, we have

$$\begin{aligned} &-(1-4p_1q_1)\log(1-4p_1q_1) - (1-4p_2q_2)\log(1-4p_2q_2) \\ &\qquad\qquad + (1-4pq)\log(1-4pq) \leqslant 64p_1q_1p_2q_2. \end{aligned} \tag{5.20}$$

Next, we simplify the second terms:

$$\begin{aligned} pq &= (p_1q_2+p_2q_1)(p_1p_2+q_1q_2) = p_1q_1(p_2^2+q_2^2) + p_2q_2(p_1^2+q_1^2) \\ &= p_1q_1(1-2p_2q_2) + p_2q_2(1-2p_1q_1) \\ &= p_1q_1 + p_2q_2 - 4p_1q_1p_2q_2. \end{aligned}$$

Hence, we have

$$-8p_1q_1 - 8p_2q_2 + 8pq = -32p_1q_1p_2q_2. \tag{5.21}$$

Finally, the first terms:

$$\begin{aligned} \mathrm{H}(p) &= \mathrm{H}(p_1q_2+p_2q_1) \\ &= (p_1q_2+p_2q_1)\log\frac{1}{(p_1q_2+p_2q_1)} + (p_1p_2+q_1q_2)\log\frac{1}{(p_1p_2+q_1q_2)} \\ &= p_1q_2\log\frac{1}{p_1q_2} + p_1q_2\log\frac{p_1q_2}{(p_1q_2+p_2q_1)} + p_2q_1\log\frac{1}{p_2q_1} + p_2q_1\log\frac{p_2q_1}{(p_1q_2+p_2q_1)} \\ &\quad + \text{ similar terms for the second summand} \\ &= q_2(-p_1\log p_1) + p_1(-q_2\log q_2) + p_2(-q_1\log q_1) + q_1(-p_2\log p_2) \\ &\quad + p_1q_2\log\frac{p_1q_2}{(p_1q_2+p_2q_1)} + p_2q_1\log\frac{p_2q_1}{(p_1q_2+p_2q_1)} \end{aligned}$$

[1] Any constant $c \geqslant \frac{1}{\ln 2}$ can be used instead of 2.

$$
\begin{aligned}
&+\ \text{similar terms from the second half}\\
&= -p_1 \log p_1(q_2+p_2) - q_1 \log q_1(p_2+q_2) - p_2 \log p_2(q_1+p_1) - q_2 \log q_2(p_1+q_1)\\
&\quad + p_1q_2 \log \frac{p_1q_2}{(p_1q_2+p_2q_1)} + p_2q_1 \log \frac{p_2q_1}{(p_1q_2+p_2q_1)} + p_1p_2 \log \frac{p_1p_2}{(p_1p_2+q_1q_2)}\\
&\quad + q_1q_2 \log \frac{q_1q_2}{(p_1p_2+q_1q_2)}\\
&= \mathrm{H}(p_1) + \mathrm{H}(p_2) - (p_1q_2+p_2q_1)\,\mathrm{H}\left(\frac{p_1q_2}{(p_1q_2+p_2q_1)}\right) - (p_1p_2+q_1q_2)\,\mathrm{H}\left(\frac{p_1p_2}{(p_1p_2+q_1q_2)}\right)\\
&\leqslant \mathrm{H}(p_1) + \mathrm{H}(p_2) - 2\min\{p_1q_2, p_2q_1\} - 2\min\{p_1p_2, q_1, q_2\} \quad \text{using } \mathrm{H}(p) \geqslant 2\min\{p,q\}\\
&\leqslant \mathrm{H}(p_1) + \mathrm{H}(p_2) - 2p_1q_2p_2q_1 - 2p_1p_2q_1q_2 \quad \text{since } \min\{p,q\} \geqslant pq \text{ for } 0 \leqslant p,q \leqslant 1\\
&= \mathrm{H}(p_1) + \mathrm{H}(p_2) - 4p_1q_1p_2q_2.
\end{aligned}
$$

Hence, we have

$$-8\,\mathrm{H}(p_1) - 8\,\mathrm{H}(p_2) + 8\,\mathrm{H}(p) \leqslant -32p_1q_1p_2q_2. \tag{5.22}$$

Combing Eq. (5.20), Eq. (5.21), Eq. (5.22), and the definition of κ Eq. (5.19), we obtain

$$\kappa(p_1) + \kappa(p_2) - \kappa(p) \leqslant 0$$

and this concludes the proof. □

5.6 Real-valued functions

Using the Fourier analytic formulae for Influence we can equivalently state the Fourier Entropy-Influence conjecture as: *there exists a universal constant C such that for all $f : \{0,1\}^n \to \{+1,-1\}$,*

$$\mathbb{H}(f) \leqslant C \cdot \sum_{S \subseteq [n]} |S|\widehat{f}(S)^2.$$

In this section, we relax the Boolean-ness condition on f, and consider *real*-valued functions $f : \{0,1\}^n \to \mathbb{R}$ defined over Boolean hypercube. The notion of entropy is not defined as is on real-valued functions, but to facilitate the discussion, without loss of generality, we will assume $\sum_S \widehat{f}(S)^2 = 1$.

Here we will establish that for all f such that $\sum_S \widehat{f}(S)^2 = 1$, and for all $\delta \in (0,1]$,

$$\mathbb{H}(f) \leqslant \sum_S |S|^{1+\delta} \widehat{f}(S)^2 + (\log n)^{O(\frac{1}{\delta})}.$$

We note a useful observation.

Lemma 5.6.1. *For any t, let $\mathcal{T} \subseteq \{S \mid |\widehat{f}(S)| \leqslant 1/t\}$. Suppose $|\mathcal{T}| \leqslant t$. Then,*

$$\sum_{S \in \mathcal{T}} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant 2.$$

Furthermore, for any k,

$$\sum_{S : |S| \leqslant k} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant 2 + 2k \log n.$$

Proof. We will prove the second part of the lemma since that includes proof of the first part. First note that the number of summands in the second part is at most n^k. Let $S_k := \{S \mid |\widehat{f}(S)| < 1/n^k\}$, then

$$\sum_{S \in S_k} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant \frac{2}{n^k} \sum_{S \in S_k} |\widehat{f}(S)| \log\left(\frac{1}{|\widehat{f}(S)|}\right) \leqslant 2,$$

where the last inequality follows from the fact that $|\widehat{f}(S)| \log(1/|\widehat{f}(S)|) < 1$, since $x \log(1/x) < 1$, for all $0 \leqslant x \leqslant 1$.

Now for all S such that $|S| \leqslant k$ and $S \notin S_k$, $\log(1/|\hat{f}(S)|) \leqslant k \log n$. Hence,

$$\sum_{S : |S| \leqslant k \text{ and } S \notin S_k} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant 2k \log n.$$

□

Theorem 5.6.2. *If* $f = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S$ *is a real-valued function on the domain* $\{0,1\}^n$ *such that* $\sum_S \widehat{f}(S)^2 = 1$, *then, for any* $\delta \in (0,1]$,

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant \sum_{S\subseteq[n]} |S|^{1+\delta}\widehat{f}(S)^2 + 2(2\log n)^{\frac{1+\delta}{\delta}} + O(\log\log n/\log(1+\delta)).$$

Proof. Since the proof consists of careful counting, we highlight our proof strategy first. We partition the Fourier coefficients into suitable parts and then upper bound each part. We start with suitably chosen sets $A_0, B_0 \subseteq 2^{[n]}$ and then inductively construct the sets $A_1, B_1, \ldots, A_k, B_k$. The A_i's represent the new Fourier coefficients whose total entropy we are able to upper bound. The B_i's represent the Fourier coefficients that are *not yet accounted for*. Our construction yields that as k increases B_k only consists of those $\widehat{f}(S)$ for which $|S| < \psi(k,n,\delta)$, where ψ is a suitable function of k, n and δ. Finally an appropriate choice of k gives us the desired inequality.

Following this strategy, we start by describing the sets A_i and B_i.

Let A_0 be the set of all $S \subseteq [n]$ for which $|S|^{1+\delta}$ is at least $\log\left(\frac{1}{\widehat{f}(S)^2}\right)$. That is,

$$A_0 := \{S \mid \widehat{f}(S)^2 \geqslant 1/2^{|S|^{1+\delta}}\}.$$

Clearly,

$$\sum_{S\in A_0} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant \sum_{S\in A_0} |S|^{1+\delta}\widehat{f}(S)^2.$$

Now, let A_1 be all the $S \subseteq [n]$ for which $|\widehat{f}(S)| < 2^{-n}$. Since $|A_1|$ is clearly at most 2^n, Lemma 5.6.1 above applies and we conclude that

$$\sum_{S\in A_1} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant 2.$$

Further let $B_1 = \{0,1\}^n \setminus (A_0 \cup A_1)$. By the definition of A_0 and A_1,

$$B_1 \subseteq \left\{ S \mid \frac{1}{2^{2n}} \leqslant \widehat{f}(S)^2 \leqslant \frac{1}{2^{|S|^{1+\delta}}} \right\}.$$

It follows that $B_1 \subseteq \{S \mid |S| \leqslant (2n)^{1/(1+\delta)}\}$. Let $r_1 := (2n)^{1/(1+\delta)}$. Thus, $|B_1| \leqslant \sum_{i=0}^{r_1} \binom{n}{i} < n^{r_1}$.

Next, let $A_2 := \{S \in B_1 : |\widehat{f}(S)| \leqslant 1/n^{r_1}\}$ and $B_2 := B_1 \setminus A_2$.

First, note that, since $A_2 \subseteq B_1$ and $|B_1| \leqslant n^{r_1}$, Lemma 5.6.1 can be applied to A_2 and hence the contribution of coefficients from A_2 is at most 2.

We also have,

$$B_2 \subseteq \left\{ S \mid \frac{1}{n^{2r_1}} \leqslant \widehat{f}(S)^2 \leqslant \frac{1}{2^{|S|^{1+\delta}}} \right\}.$$

Let $r_2 = (\log(n^{2r_1}))^{1/(1+\delta)} = (2r_1 \log n)^{1/(1+\delta)}$. It is then clear that for $S \in B_2$, we must have $|S| \leqslant r_2$ and thus $|B_2| \leqslant n^{r_2}$.

Continuing this way, we define

$$\begin{aligned} r_{k+1} &:= (2r_k \log n)^{1/(1+\delta)}, \\ A_{k+1} &:= \{S \in B_k \mid |\widehat{f}(S)| \leqslant 1/n^{r_k}\}, \text{ and} \\ B_{k+1} &:= B_k \setminus A_{k+1}. \end{aligned}$$

In general, then,

$$B_{k+1} \subseteq \left\{ S \mid \frac{1}{n^{2r_k}} \leqslant \widehat{f}(S)^2 \leqslant \frac{1}{2^{|S|^{1+\delta}}} \right\}.$$

Thus $B_{k+1} \subseteq \{S \mid |S| \leqslant r_{k+1}\}$, and so, $|B_{k+1}| \leqslant n^{r_{k+1}}$. Since $A_{k+1} \subseteq B_k$, $|A_{k+1}| \leqslant n^{r_k}$ and Lemma 5.6.1 can be applied to A_{k+1}.

It is easy to see by induction that for $k \geqslant 1$,

$$r_k = (2\log n)^{\frac{1}{\delta}\left(1-(1+\delta)^{-k+1}\right)} \cdot (2n)^{(1+\delta)^{-k}}.$$

Thus, $r_k \leqslant (2\log n)^{\frac{1}{\delta}} \cdot (2n)^{(1+\delta)^{-k}}$.

By taking $k^* := \log\log 2n/\log(1+\delta)$, we get $r_{k^*} \leqslant 2(2\log n)^{\frac{1}{\delta}}$.

We repeat the above process up to k^* times. For each $k \leqslant k^*$, the coefficients from A_k contribute at most 2 to the entropy by the first part of the lemma. Note that for all sets $S \in B_{k^*}$, $|S| \leqslant r_{k^*}$. For $k = k^*$, we apply the second part of proof of Lemma 5.6.1 and conclude that coefficients from B_{k^*} contribute at most $2r_{k^*}\log n \leqslant 2\cdot(2\log n)^{1+\frac{1}{\delta}}$. Moreover, note that $A_0 \cup A_1 \cup \cdots \cup A_{k^*} \cup B_{k^*}$ is a cover of $2^{[n]}$. Hence, we accounted for contributions to the entropy from all coefficients.

Altogether, we get the total entropy to be at most

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) \leqslant \sum_{S\subseteq[n]} |S|^{1+\delta}\widehat{f}(S)^2 + 2\log\log 2n/\log(1+\delta) + 2(2\log n)^{1+\frac{1}{\delta}}.$$

$\square$

A corollary of Theorem 5.6.2 is an upper bound on the Fourier Entropy of a real-valued function in terms of the first and second moments of the sensitivities of the function.

Corollary 5.6.3. *If $f = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S$ is a real-valued function on the domain $\{0,1\}^n$ such that $\sum_S \widehat{f}(S)^2 = 1$, then, for any $\delta \in (0,1]$,*

$$\sum_{S\subseteq[n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) = \mathsf{as}(f)^{1-\delta}\mathsf{as}_2(f)^{\delta} + 2(2\log n)^{\frac{1+\delta}{\delta}} + O(\log\log n/\log(1+\delta)),$$

where $\mathsf{as}_2(f) := \sum_S |S|^2\widehat{f}(S)^2$.

Note that in the above statements $\mathsf{as}(f)$ is defined via its Fourier expansion, that is, $\mathsf{as}(f) := \sum_S |S|\widehat{f}(S)^2$. Similarly, $\mathsf{as}_2(f)$, in spite of having a combinatorial definition (see Proposition 5.6.4), is defined to be $\sum_S |S|^2\widehat{f}(S)^2$.

The proof of Corollary 5.6.3 is straightforward from the following lemma. For the proof of the lemma we need the following proposition which is well-known; see for instance [GPS10, Eq. 2.11] or [O'D14, Ex. 2.20].

Proposition 5.6.4 ([GPS10]). *For* $f : \{0,1\}^n \to \{+1,-1\}$,

$$\frac{1}{2^n}\sum_x \mathsf{s}_f(x)^2 = \sum_{S\subseteq[n]} |S|^2 \widehat{f}(S)^2 = \mathsf{as}_2(f).$$

Lemma 5.6.5. *Let* $f : \{0,1\}^n \to \mathbb{R}$, *and* $0 \leqslant \delta \leqslant 1$. *Then,*

$$\sum_{S\subseteq[n]} |S|^{1+\delta}\widehat{f}(S)^2 \leqslant \mathsf{as}(f)^{1-\delta}\mathsf{as}_2(f)^{\delta}.$$

Proof. For $\delta = 0$, this is the Fourier expression for average sensitivity. For $\delta = 1$, this is Proposition 5.6.4. We next prove it for $\delta = 1/2$. We treat $\widehat{f}(S)^2$ as the probability associated to the set S and use the following version of the Cauchy-Schwartz inequality: for any two random variables $X, Y : \Omega \to \mathbb{R}$, we have $\mathbb{E}(XY) \leqslant \sqrt{\mathbb{E}(X^2)}\sqrt{\mathbb{E}(Y^2)}$. Choosing $X(S) = \sqrt{|S|}$ and $Y(S) = |S|$ immediately yields the desired inequality for the value of $\delta = \frac{1}{2}$ in light of Proposition 5.6.4.

In general, we can show the following: if the desired inequality holds for $\delta = \alpha$ and $\delta = \beta$ then the inequality must also hold for $\delta = \frac{\alpha+\beta}{2}$. To show this, one may apply the Cauchy-Schwartz inequality with $X(S) = |S|^{(1+\alpha)/2}$ and $Y(S) = |S|^{(1+\beta)/2}$.

Hence, by continuity, the desired inequality holds for any $\delta \in [0,1]$. □

5.7 Examples

In this section we give examples of non-Boolean functions with large Fourier entropy.

A decision tree for a non-Boolean, say $\mathbb{R}$-valued, function f can be defined by a natural generalisation of the definition for a Boolean-valued function. It queries the (Boolean) input variables as in the usual decision tree, but produces a value in $\mathbb{R}$ at each leaf. It must guarantee that on all inputs that reach a leaf the function value must be constant and equal to the value produced at that leaf.

The first example shows that (in contrast to Inequality (5.3) for Boolean functions) the Fourier entropy cannot be upper bounded by log(#leaves) for non-Boolean f. In fact, there is an exponential gap:

Lemma 5.7.1. *There exists a function $f : \{0,1\}^n \to \mathbb{R}$ satisfying $\sum_S \widehat{f}(S)^2 = 1$ such that*

$$\sum_{S \subseteq [n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) = \Omega(n), \quad \text{but} \quad \log L(f) = O(\log n).$$

Proof. Consider the following function:

$$f(x) = \sqrt{\frac{2^{d(x)}}{n+2}},$$

where $d(x) = n+1$, if $x = 0^n$, else it is the first index in x that is 1. Note that this function has a decision tree same as the OR function and thus have only $n+1$ leaves. Now to see that $\sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1$ consider the following:

$$\sum_x f(x)^2 = \sum_{i \in [n+1]} \sum_{x : d(x) = i} f(x)^2 = \sum_{i \in [n]} 2^{n-i} \frac{2^i}{n+2} + \frac{2^{n+1}}{n+2} = 2^n,$$

and thus from Parseval's identity we have $\sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1$.

It is easy to check that for any set $S \subseteq [n]$ if k is the largest index in S then

$$|\widehat{f}(S)| = \frac{1}{2^n}\left(2^{n-k}\sqrt{\frac{2^k}{n+2}} - \sum_{i=k+1}^{n} 2^{n-i}\sqrt{\frac{2^i}{n+2}} - \sqrt{\frac{2^{n+1}}{n+2}}\right) \approx \frac{1}{\sqrt{n2^k}}.$$

And from this it follows that the entropy for the Fourier coefficient squares is around $n/2 + \log n$ whereas $\log(L(f)) = \log(n)$. □

The next example shows that (in contrast to Inequality (5.3) for Boolean functions) Fourier entropy can be logarithmically larger than the degree for non-Boolean functions. It also shows a logarithmic gap between influence and Fourier entropy.

Lemma 5.7.2. *There exists a function $f : \{0,1\}^n \to \mathbb{R}$ of degree d satisfying $\sum_S \widehat{f}(S)^2 = 1$ such that*

$$\sum_{S \subseteq [n]} \widehat{f}(S)^2 \log\left(\frac{1}{\widehat{f}(S)^2}\right) = \Omega(d \log n).$$

Proof. Consider the following function $f = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S$, where $\widehat{f}(S) = 1/\sqrt{\binom{n}{2}}$ if $|S| = 2$, and $\widehat{f}(S) = 0$ otherwise. It is easy to see that the $\mathbb{H}(f) = \log\binom{n}{2}$, whereas $\mathsf{Inf}(f) = \sum_{S\subseteq[n]} |S|\widehat{f}(S)^2 = 2$.

So now if we put uniform weights on k-sized sets, that is, $\widehat{f}(S) = 1/\sqrt{\binom{n}{k}}$ if $|S| = k$, and $\widehat{f}(S) = 0$ if $|S| \neq k$, we will get $\mathsf{Inf}(f) = k$ and $\mathbb{H}(f) = \log\binom{n}{k} \geqslant k\log n - k\log k$. Choosing $k = \sqrt{n}$, we will have $\mathbb{H}(f) = \Omega(\sqrt{n}\log n)$ and $\mathsf{Inf}(f) = \sqrt{n}$. Since the degree of the function is $d = \sqrt{n}$, we get $\mathbb{H}(f) = \Omega(d \cdot \log n)$. Also, $\mathbb{H}(f) = \Omega(\mathsf{Inf}(f) \cdot \log n)$ □

5.8 Conclusion

In this chapter, we studied a particular problem called the *Fourier Entropy-Influence Conjecture*. Like many other problems (e.g. sensitivity vs block-sensitivity [NS94]) in Fourier analysis of Boolean functions, the FEI conjecture also remains wide open. There are plenty of questions that remain open, we mention a few here:

- *Can we upper bound the Fourier-entropy of a Boolean function by the combinatorial measures that bound the influence (see Fig. 5.1), like sensitivity, block sensitivity, certificate complexity, average certificate complexity, etc.?* Recently, in an ongoing work with Michal Koucký, we have been able to establish that the Fourier entropy of f is at most the *subcube partition entropy* of a partition that computes f. This improves on Theorem 5.3.11.

- Proving the FEI conjecture for special classes of Boolean functions has turned out to be a non-trivial task. The proofs, for the classes where we know the conjecture is true, have used varied techniques. Specifically, *Can we prove the conjecture for linear threshold functions, or monotone functions?*

- Can we prove the following seemingly weaker-looking version of the conjecture: *Does there exists a universal constant C such that for all* $f\colon \{0,1\}^n \to \{+1,-1\}$,

$$\min_{S\subseteq[n]} \log \frac{1}{\widehat{f}(S)^2} \leqslant C \cdot \mathsf{Inf}(f) \ ?$$

- It is known that strong enough Fourier-concentration is sufficient to yield the FEI conjecture (Bourgain-Kalai, see [KMS12, Theorem 3.4]). *Can we go in the reverse direction?* That is, assuming the FEI conjecture holds for a class of functions, can we obtain concentration inequality (for the distribution given by the Fourier spectrum) for this class of functions?

Chapter 6

Conclusion

As we mentioned in the Introduction, this thesis has two parts, namely Algebraic complexity theory and Boolean function analysis.

Part I

In the first part our aim has been to understand the landscape around VP in the hope that this will shed some light on the difficult problem of proving general lower bounds, i.e., separating algebraic classes *unconditionally*.

In Chapter 2, we studied different kinds of reductions in the algebraic setting while establishing that every family in VNP can be written as a difference of two projections of sym-Perm. In other words, we showed that sym-Perm is VNP-complete with respect to linear p-projections over fields of characteristic not equal to 2. It remains open *whether* sym-Perm *is* VNP*-complete with respect to p-projections?* We, then, proved lower bounds against monotone projections. In particular, we showed that $\mathsf{Clique}^{\sqrt{n}}$ is not a monotone p-projection of Perm. This rules out a technique that aims to prove better lower bounds for Perm_n by transferring lower bound from Clique to Perm via projections. We also studied the closure property of (multilinear) algebraic classes under exponential sums.

Further, in Chapter 3, we defined and studied *homomorphism polynomials*. Using homomorphism polynomials we characterised the algebraic classes VBP, VP, and VNP. In particular, we established the first instance of natural families of polynomials that are VP-complete. Motivated by the characterisation of circuit classes by restrictions on tree-width of the homomorphism polynomials, we were naturally led to the study of families of polynomials with *intermediate* complexity.

We discussed families of polynomials with intermediate complexity in Chapter 4. Here we established a list of new VNP-intermediate polynomial families. Moreover, the definitions of the intermediate families are based on basic (combinatorial) NP-complete problems.

Several interesting questions remain open. We note a few here. (They are also mentioned at the end of Chapters 2, 3, and 4.)

- *What is the complexity of homomorphism polynomials that are defined on a family G_n such that G_n has tree-width* $o(n)$*?*
- *Find new natural families that are* VP*-complete.*
- *Can we find families of polynomials, with integer coefficients, that are* VNP*-intermediate over all finite fields?, or fields with non-zero characteristic?, or characteristic zero?*
- *Are there polynomial families of intermediate complexity between* VBP *and* VP*?*

Part II

In the second part, we moved to Fourier analysis of Boolean functions. Here our aim has been to try and prove a longstanding open problem called the *Fourier Entropy-Influence Conjecture* for restricted classes of Boolean functions, or study (weaker) variants of the conjecture in the hope that this sheds some light on how to tackle the conjecture in full generality.

In Chapter 5, we first established upper bounds on Fourier entropy of a Boolean function in terms of combinatorial measures, like average depth of a decision tree, etc., that are known to bound the influence of a function from above (see Fig. 5.1). We then showed that the Fourier entropy of a linear threshold function on n-variables is $O(\sqrt{n})$. We further generalise the proof technique to obtain similar bounds on the Fourier entropy of a degree-d polynomial threshold function. (The upper bound matches the best known bound on the influence of polynomial threshold functions [HKM14, DRST14].) Next, using "tensorizability" properties of Fourier entropy and Influence, we proved the Fourier Entropy-Influence conjecture for *Read-Once* formulas over AND, OR, NOT, and XOR gates. Finally we established an upper bound, resembling the Fourier-analytic formulae for average sensitivity, on entropy of real-valued functions.

The Fourier Entropy-Influence conjecture itself remains wide open. We state a few interesting open questions here that may help us make progress towards the general conjecture. (They are also mentioned at the end of Chapter 5.)

- *Can we prove the conjecture for special classes of Boolean functions, for example, linear threshold functions, or monotone functions?* It seems this case itself presents us with non-trivial obstacles to overcome, because the FEI conjecture implies the famed KKL Theorem [KKL88] (see [OWZ11]) for which we know no proof that avoids *hypercontractivity*, or *log-Sobolev inequality*.

- *Can we upper bound the Fourier-entropy of a Boolean function by the combinatorial measures that bounds the influence, like sensitivity, block sensitivity, certificate complexity, average certificate complexity, etc.?*

- *Does there exists a universal constant C such that for all $f\colon \{0,1\}^n \to \{+1,-1\}$,*

$$\min_{S\subseteq[n]} \log \frac{1}{\widehat{f}(S)^2} \leqslant C \cdot \mathsf{Inf}(f) \ ?$$

Bibliography

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998.

[AT13] David Avis and Hans Raj Tiwary. On the Extension Complexity of Combinatorial Polytopes. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP Part I*, pages 57–68, 2013.

[AZ90] R. Ahlswede and Z. Zhang. An Identity in Combinatorial Extremal Theory. *Advances in Mathematics*, 80(2):137 – 151, 1990.

[BCL+06] Christian Borgs, Jennifer Chayes, László Lovász, Vera T. Sós, and Katalin Vesztergombi. Counting Graph Homomorphisms. In *Topics in Discrete Mathematics*, volume 26 of *Algorithms and Combinatorics*, pages 315–371. Springer Berlin Heidelberg, 2006.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity Measures and Decision Tree Complexity: A Survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[Bea00] Arnaud Beauville. Determinantal Hypersurfaces. *The Michigan Mathematical Journal*, 48(1):39–64, 2000.

[BK97] Jean Bourgain and Gil Kalai. Influences of Variables and Threshold Intervals under Group Symmetries. *Geometric and Functional Analysis (GAFA)*, 7(3):438–461, 1997.

[BKS99] Itai Benjamini, Gil Kalai, and Oded Schramm. Noise Sensitivity of Boolean Functions and Applications to Percolation. *Publications Mathematiques de l'Institut des Hautes Etudes Scientifiques*, 90(1):5–43, 1999.

[BL85] Michael Ben-Or and Nathan Linial. Collective Coin Flipping, Robust Voting Schemes and Minima of Banzhaf Values. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 408–416, 1985.

[Blä01] Markus Bläser. Complete Problems for Valiant's Class of qp-Computable Families of Polynomials. In *Computing and Combinatorics, 7th Annual International Conference, COCOON 2001, Guilin, China, August 20-23, 2001, Proceedings*, pages 1–10, 2001.

[BOH90] Yigal Brandman, Alon Orlitsky, and John Hennessy. A Spectral Lower Bound Technique for the Size of Decision Trees and Two-Level AND/OR Circuits. *IEEE Transactions of Computers*, 39(2):282–287, 1990.

[Brä11] Petter Brändén. Obstructions to determinantal representability. *Advances in Mathematics*, 226(2):1202–1212, 2011.

[Brä13] Petter Brändén. Hyperbolicity cones of Elementary Symmetric Polynomials are spectrahedral. *Optimization Letters*, 8(5):1773–1782, 2013.

[Bür99] Peter Bürgisser. On the Structure of Valiant's Complexity Classes. *Discrete Mathematics & Theoretical Computer Science*, 3(3):73–94, 1999.

[Bür00a] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer, 2000.

[Bür00b] Peter Bürgisser. Cook's versus Valiant's hypothesis. *Theoretical Computer Science*, 235(1):71–88, 2000.

[Cat81] F. Catanese. Babbage's conjecture, contact of surfaces, symmetric determinantal varieties and applications. *Inventiones mathematicae*, 63(3):433–465, 1981.

[Cay69] A. Cayley. A Memoir on Quartic Surfaces. *Proceedings of the London Mathematical Society*, s1-3(1):19–69, 1869.

[CDM13] Florent Capelli, Arnaud Durand, and Stefan Mengel. The Arithmetic Complexity of Tensor Contractions. In *Symposium on Theoretical Aspects of Computer Science STACS*, volume 20 of *LIPIcs*, pages 365–376, 2013.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CKLS15] Sourav Chakraborty, Raghav Kulkarni, Satyanarayana V. Lokam, and Nitin Saurabh. *Upper Bounds on Fourier Entropy*, pages 771–782. Springer International Publishing, 2015.

[CT79] R. J. Cook and A. D. Thomas. Line Bundles and Homogeneous Matrices. *The Quarterly Journal of Mathematics*, 30(4):423–429, 1979.

[Dam91] Carsten Damm. $\mathrm{DET}=\mathrm{L}^{(\#\mathrm{L})}$. Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt–Universität zu Berlin, 1991.

[Dic21] Leonard E. Dickson. Determination of all general homogeneous polynomials expressible as determinants with linear elements. *Transactions of the Americal Mathematical Society*, 22:167–179, 1921.

[Dix02] A. C. Dixon. Note on the reduction of a ternary quantic to a symmetrical determinant. *Proc. Cambridge Philos. Soc.*, 11:350–351, 1902.

[DLM+07] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 549–558, 2007.

[DMM+16] Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism Polynomials Complete for VP. *Chicago Journal of Theoretical Computer Science*, 2016(3), March 2016.

[DMPY12] Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating Multilinear Branching Programs and Formulas. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 615–624. ACM, 2012.

[dRA12] Nicolas de Rugy-Altherre. A Dichotomy Theorem for Homomorphism Polynomials. In *Mathematical Foundations of Computer Science 2012*, volume 7464 of *LNCS*, pages 308–322. Springer Berlin Heidelberg, 2012.

[DRST14] Ilias Diakonikolas, Prasad Raghavendra, Rocco Servedio, and Li-Yang Tan. Average Sensitivity and Noise Sensitivity of Polynomial Threshold Functions. *SIAM Journal on Computing*, 43(1):231–253, 2014.

[dW08] Ronald de Wolf. A Brief Introduction to Fourier Analysis on the Boolean Cube. *Theory of Computing, Graduate Surveys*, 1:1–20, 2008.

[ER60] Paul Erdös and Alfréd Rényi. On the Evolution of Random Graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.

[FK96] Ehud Friedgut and Gil Kalai. Every Monotone Graph Property has a Sharp Threshold. *Proceedings of the American Mathematical Society*, 124(10):2993–3002, 1996.

[FK97] Uriel Feige and Joe Kilian. On Limited versus Polynomial Nondeterminism. *Chicago Journal of Theoretical Computer Science*, 1997(1), March 1997.

[FMP+15] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Exponential Lower Bounds for Polytopes in Combinatorial Optimization. *Journal of the ACM*, 62(2):17, 2015.

[Fri98] Ehud Friedgut. Boolean Functions with Low Average Sensitivity Depend on Few Coordinates. *Combinatorica*, 18(1):27–35, 1998.

[FvzGR86] F. Fich, J. von zur Gathen, and C. Rackoff. Complete Families of Polynomials. Manuscript, 1986.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GKK08] Parikshit Gopalan, Adam Tauman Kalai, and Adam Klivans. Agnostically Learning Decision Trees. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 527–536, 2008.

[GKKP11] Bruno Grenet, Erich L Kaltofen, Pascal Koiran, and Natacha Portier. Symmetric determinantal representation of formulas and weakly skew circuits. *Randomization, Relaxation, and Complexity in Polynomial Equation Solving*, 556 of Contemporary Mathematics - American Mathematical Society:61–96, 2011.

[GL94] Craig Gotsman and Nathan Linial. Spectral Properties of Threshold Functions. *Combinatorica*, 14(1):35–50, 1994.

[GMT13] Bruno Grenet, Thierry Monteil, and Stéphan Thomassé. Symmetric Determinantal Representations in characteristic 2. *Linear Algebra and its Applications*, 439(5):1364–1381, 2013.

[GPS10] Christophe Garban, Gabor Pete, and Oded Schramm. The Fourier Spectrum of Critical Percolation. *Acta Mathematica*, 205:19–104, 2010.

[Gra55] H. Grassmann. Die stereometrischen Gleichungen dritten Grades, und die dadurch erzeugten Oberflächen. *Journal für die reine und angewandte Mathematik*, 1855:47–65, 1855.

[Gro15] Joshua A. Grochow. Monotone projection lower bounds from Extended Formulation lower bounds. ECCC Tech. Report TR15-171 and arXiv:1510.08417 [cs.CC], 2015.

[Hes55] Otto Hesse. Über Determinanten und ihre Anwendung in der Geometrie, insbesondere auf Curven vierter Ordnung. *Journal für die reine und angewandte Mathematik*, 1855:243–264, 1855.

[HKM14] Prahladh Harsha, Adam Klivans, and Raghu Meka. Bounding the Sensitivity of Polynomial Threshold Functions. *Theory of Computing*, 10(1):1–26, 2014.

[HMV06] J. William Helton, Scott A. McCullough, and Victor Vinnikov. Noncommutative convexity arises from linear matrix inequalities. *Journal of Functional Analysis*, 240(1):105–191, 2006.

[HN04] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford lecture series in mathematics and its applications. Oxford University Press, 2004.

[Hru15] Pavel Hrubeš. On Hardness of Multilinearization, and VNP Completeness in Characteristics Two. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:67, 2015.

[HV07] J. William Helton and Victor Vinnikov. Linear Matrix Inequality Representation of Sets. *Communications on Pure and Applied Mathematics*, 60(5):654–674, 2007.

[HWY10] Pavel Hrubeš, Avi Wigderson, and Amir Yehudayoff. Relationless Completeness and Separations. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010*, pages 280–290, 2010.

[HY11] Pavel Hrubeš and Amir Yehudayoff. Arithmetic Complexity in Ring Extensions. *Theory of Computing*, 7(1):119–129, 2011.

[JKRS09] Ali Juma, Valentine Kabanets, Charles Rackoff, and Amir Shpilka. The Black-Box Query Complexity of Polynomial Summation. *Computational Complexity*, 18(1):59–79, 2009.

[JMR13] Maurice Jansen, Meena Mahajan, and B.V.Raghavendra Rao. Resource Trade-offs in Syntactically Multilinear Arithmetic Circuits. *Computational Complexity*, 22(3):517–564, 2013.

[JR09] Maurice Jansen and B.V. Raghavendra Rao. Simulation of Arithmetical Circuits by Branching Programs with Preservation of Constant Width and Syntactic Multilinearity. In *Computer Science - Theory and Applications*, volume 5675 of *Lecture Notes in Computer Science*, pages 179–190. Springer Berlin Heidelberg, 2009.

[JS82] Mark Jerrum and Marc Snir. Some Exact Complexity Results for Straight-Line Computations over Semirings. *Journal of the ACM*, 29(3):874–897, 1982.

[Juk14] Stasys Jukna. Why is Hamilton Cycle so different from Permanent? http://cstheory.stackexchange.com/questions/27496/why-is-hamiltonian-cycle-so-different-from-permanent, 2014.

[Kal] Gill Kalai. The Entropy/Influence Conjecture. Terence Tao's blog.

[Kal86] Erich Kaltofen. Uniform Closure Properties of P-computable Functions. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 330–337, 1986.

[Kal87] Erich Kaltofen. Single-factor Hensel Lifting and Its Application to the Straight-Line Complexity of Certain Polynomials. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 443–452, 1987.

[Kal89] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. In *Randomness and Computation*, volume 5 of *the Advances in Computing Research series*, pages 375–412. JAI Press Inc., Greenwich CT, 1989.

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The Influence of Variables on Boolean Functions. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 68–80, 1988.

[KL82] Richard M Karp and Richard Lipton. Turing machines that take advice. *Lą́renseignement mathématique*, 28(2):191–209, 1982.

[KLW10] Adam Klivans, Homin Lee, and Andrew Wan. Mansour's Conjecture is True for Random DNF Formulas. In *Proceedings of the 23rd Conference on Learning Theory*, pages 368–380, 2010.

[KMS12] Nathan Keller, Elchanan Mossel, and Tomer Schlank. A note on the Entropy/Influence conjecture. *Discrete Mathematics*, 312(22):3364 – 3372, 2012.

[KS93] Ephraim Korach and Nir Solel. Tree-width, Path-width, and Cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993.

[KT90] Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990.

[Lad75] Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, 22(1):155–171, 1975.

[LLTY15] Chia-Jung Lee, Satyanarayana V. Lokam, Shi-Chun Tsai, and Ming-Chuan Yang. Restrictions of Nondegenerate Boolean Functions and Degree Lower Bounds over different Rings. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 501–505, 2015.

[Mah14] Meena Mahajan. Algebraic complexity classes. In *Perspectives in Computational Complexity*, volume 26 of *Progress in Computer Science and Applied Logic*, pages 51–75. Springer International Publishing, 2014.

[Man95] Yishay Mansour. An $O(\mathrm{n}^{\log\log n})$ Learning Algorithm for DNF under the Uniform Distribution. *Journal of Computer and System Sciences*, 50(3):543–550, 1995.

[Mar74] Grigorii Aleksandrovich Margulis. Probabilistic Characteristics of Graphs with Large Connectivity. *Probl. Peredachi Inf.*, 10:101–108, 1974.

[Men11] Stefan Mengel. Characterizing Arithmetic Circuit Classes by Constraint Satisfaction Problems. In *Automata, Languages and Programming*, volume 6755 of *LNCS*, pages 700–711. Springer Berlin Heidelberg, 2011.

[Min78] Henryk Minc. *Permanents*. Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Company, 1978.

[MM60] T. Muir and W.H. Metzler. *A Treatise on the Theory of Determinants*. Dover Publications, 1960.

[MM61] Marvin Marcus and Henryk Minc. On the relation between the determinant and the permanent. *Illinois Journal of Mathematics*, 5(3):376–381, 1961.

[MP08] Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *Journal of Complexity*, 24(1):16–38, 2008.

[MR08] Meena Mahajan and B.V. Raghavendra Rao. Arithmetic Circuits, Syntactic Multilinearity, and the Limitations of Skew Formulae. In *Mathematical Foundations of Computer Science 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 455–466. Springer Berlin Heidelberg, 2008.

[MS16] Meena Mahajan and Nitin Saurabh. Some Complete and Intermediate polynomials in Algebraic Complexity Theory. In *Proceedings of the 11th Computer Science Symposium in Russia (CSR), 2016*, 2016.

[MST16] Meena Mahajan, Nitin Saurabh, and Sébastien Tavenas. VNP=VP in the multilinear world. *Information Processing Letters*, 116(2):179–182, 2016.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chicago Journal of Theoretical Computer Science*, 1997(5), 1997.

[MVW04] Pierre McKenzie, Heribert Vollmer, and Klaus W. Wagner. Arithmetic Circuits and Polynomial Replacement Systems. *SIAM Journal on Computing*, 33(6):1513–1531, 2004.

[NPT13] Tim Netzer, Daniel Plaumann, and Andreas Thom. Determinantal representations and the Hermite matrix. *The Michigan Mathematical Journal*, 62(2):407–420, 2013.

[NS94] Noam Nisan and Mario Szegedy. On the Degree of Boolean Functions as Real Polynomials. *Computational Complexity*, 4:301–313, 1994.

[NT12] Tim Netzer and Andreas Thom. Polynomials with and without determinantal representations. *Linear Algebra and its Applications*, 437(7):1579 – 1595, 2012.

[O'D03] Ryan O'Donnell. *Computational Applications of Noise Sensitivity*. PhD thesis, MIT, 2003.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[Oli15] Rafael Oliveira. Factors of Low Individual Degree Polynomials. In *30th Conference on Computational Complexity, CCC 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 198–216, 2015.

[O'N71] Patrick E. O'Neil. Hyperplane Cuts of an n-Cube. *Discrete Mathematics*, 1(2):193 – 195, 1971.

[OT13] Ryan O'Donnell and Li-Yang Tan. A Composition Theorem for the Fourier Entropy-Influence Conjecture. In *Proceedings of Automata, Languages and Programming - 40th International Colloquium*, pages 780–791, 2013.

[OWZ11] Ryan O'Donnell, John Wright, and Yuan Zhou. The Fourier Entropy-Influence Conjecture for certain classes of Boolean Functions. In *Proceedings of Automata, Languages and Programming - 38th International Colloquium*, pages 330–341, 2011.

[Poi08] Bruno Poizat. À la recherche de la définition de la complexité d'espace pour le calcul des polynômes à la manière de Valiant. *Journal of Symbolic Logic*, 73(4):1179–1201, 2008.

[Pol13] G. Polya. Aufgabe 424. *Archiv der Mathematik und Physik (3)*, 20:271, 1913.

[PSV11] Daniel Plaumann, Bernd Sturmfels, and Cynthia Vinzant. Quartic curves and their bitangents. *Journal of Symbolic Computation*, 46(6):712–733, 2011.

[Qua12] Ronan Quarez. Symmetric Determinantal Representation of Polynomials. *Linear Algebra and its Applications*, 436(9):3642–3660, 2012.

[Raz85a] A. A. Razborov. Lower bounds on monotone complexity of the logical Permanent. *Mathematical notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985.

[Raz85b] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.

[Raz06] Ran Raz. Separation of Multilinear Circuit and Formula Size. *Theory of Computing*, 2(6):121–135, 2006.

[Raz10] Ran Raz. Elusive Functions and Lower Bounds for Arithmetic Circuits. *Theory of Computing*, 6:135–177, 2010.

[Rot14] Thomas Rothvoß. The Matching Polytope has exponential Extension Complexity. In *Symposium on Theory of Computing (STOC), 2014*, pages 263–272, 2014.

[Rus81] Lucio Russo. On the Critical Percolation Probabilities. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 56(2):229–237, 1981.

[Sch81] Friedrich Schur. Ueber die durch collineare Grundgebilde erzeugten Curven und flächen. *Mathematische Annalen*, 18(1):1–32, 1881.

[SS77] Eli Shamir and Marc Snir. Lower bounds on the number of multiplications and the number of additions in the monotone computations. Technical Report Technical Report IBM RC 6757, IBM, 1977.

[SS80] Eli Shamir and Marc Snir. On the Depth Complexity of Formulas. *Mathematical Systems Theory*, 13(1):301–322, 1980.

[Str73] Volker Strassen. Vermeidung von Divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Sze13] G. Szegö. Losung zu 424. *Archiv der Mathematik und Physik (3)*, 21:291–292, 1913.

[Tod92] Seinosuke Toda. Classes of Arithmetic Circuits capturing the Complexity of computing the Determinant. *IEICE Transactions on Information and Systems*, E75-D:116–124, 1992.

[TT94] Prasoon Tiwari and Martin Tompa. A Direct Version of Shamir and Snir's Lower Bounds on Monotone Circuit Depth. *Information Processing Letters*, 49(5):243–248, 1994.

[Val79] Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 249–261, 1979.

[Val82] Leslie G. Valiant. Reducibility by algebraic Projections. In *Logic and Algorithmic: International Symposium in honour of Ernst Specker*, volume 30, pages 365–380. Monograph. de l'Enseign. Math., 1982.

[Val92] Leslie G. Valiant. Why is Boolean Complexity Theory Difficult? In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 84–94, 1992.

[Ven92] H. Venkateswaran. Circuit Definitions of Nondeterministic Complexity Classes. *SIAM Journal on Computing*, 21(4):655–670, 1992.

[Vin91] V. Vinay. Counting Auxiliary Pushdown Automata and Semi-Unbounded Arithmetic Circuits. In *Proceedings of the 6th Structure in Complexity Theory Conference*, volume 223 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 1991.

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.

[VT89] H. Venkateswaran and Martin Tompa. A New Pebble Game that Characterizes Parallel Complexity Classes. *SIAM Journal on Computing*, 18(3):533–549, 1989.

[vzG87] Joachim von zur Gathen. Feasible arithmetic computations: Valiant's hypothesis. *Journal of Symbolic Computation*, 4(2):137–172, 1987.

[WWW14] Andrew Wan, John Wright, and Chenggang Wu. Decision Trees, Protocols and the Entropy-Influence Conjecture. In *Innovations in Theoretical Computer Science, ITCS'14*, pages 67–80, 2014.