SAPIENZA
UNIVERSITÀ DI ROMA

La Sapienza - Università di Roma

Dottorato di Ricerca in Informatica

XXI Ciclo – 2008

Degree Lower bounds for Algebraic Proof Systems

Massimo Lauria

La Sapienza - Università di Roma

Dottorato di Ricerca in Informatica

XXI Ciclo - 2008

Massimo Lauria

Degree Lower bounds for Algebraic Proof Systems

Thesis Committee		Reviewers
Prof. Nicola Galesi	(Advisor)	Prof. Juan Luis Esteban
Prof. Angelo Monti		Prof. Stefan Dantchev
Prof. Alessandro Mei		

Author's address:

Massimo Lauria
Dipartimento di Informatica
Sapienza, Università di Roma
Via Salaria 113, 00198 Roma, Italy
e-mail: `lauria@di.uniroma1.it`
www: `http://www.dsi.uniroma1.it/~lauria`

Acknowledgements

This thesis is not merely the product of its writer's effort, but also the result of the joint effort of the people around him. I wouldn't be able to achieve this result without the support of the closest members of my family. My parents Barbara and Giuseppe, my sister Mariangela and my uncle/friend Andrea. Since the days of my childhood they have "enjoyed" all the annoyances of having around someone interested in computers, computer science and math. But yet they gave me their whole support for the pursuit of my interests. I hope this thesis will convince them that now I have different people to annoy.

The first person in academia I need to thank is Angelo Monti. He introduced me to the fascinating field of Computational Complexity and to the world of professional research.

I need to mention my colleague and friend Anna de Mier who gave me advice and support in several moments of my graduate studies. It is a safe bet that I'm going to rely on her wisdom again in the future.

During my PhD program I had the pleasure to visit the Computer Science Department at the University of Toronto, where I had the honour to meet and to be lectured by many excellent scientists. Far too many to name all of them here. Among them I thank Toniann Pitassi for hosting me and for the stimulating discussions we had. I also want to thank the great students of their theory lab.

Last but not least I thank my advisor Nicola Galesi for support, help and for introducing me to many interesting and challenging problems. I hope to have met his expectancy.

Abstract

Given a propositional formula ϕ we are interested in studying how difficult is to prove that ϕ is a tautology. Multivariate polynomials can be manipulated in a specific way to obtain such a proof. We study the power of this algebraic manipulation. In particular its running time is deeply related with the degree of the polynomials involved. We prove that this method behave very badly in some cases, by showing that there are tautologies with very short proofs but which require high degree polynomials.

This does not exclude that another method of polynomial manipulation could lead to an efficient algorithm. In this case we are able to prove that if such efficient algorithm exists, it could be used to solve problems believed to be very hard.

Algebraic proof systems discussed so far are limited. We define a slightly stronger system $\mathrm{P}\textsc{cr}_k$. We give examples of tautologies that are easy for this system, but very hard for systems known in literature. Nevertheless we show that a random tautology is still very hard to prove for $\mathrm{P}\textsc{cr}_k$.

Contents

Acknowledgements **i**

Abstract **iii**

1 Introduction **1**

1.1 Propositional Proof systems 2

1.1.1 Resolution 3

1.1.2 Polynomially Calculus 4

1.1.3 Res_k, a resolution on k-DNF 6

1.2 Algebraic Representation of boolean functions 6

1.2.1 Symbolic variables for negation 7

1.3 Partial assignments and restrictions 8

1.4 Thesis's content 9

2 Proof search techniques in Polynomial Calculus **11**

2.1 Order of monomials in residue 11

2.2 Division algorithm 12

2.3 Gröbner basis 13

2.3.1 Computation of a Gröbner base 14

2.4 Bounded degree Gröbner base computation 15

2.5 Degree complexity of a proof 17

2.5.1 Scheme of degree lower bounds 18

3 Graph Ordering Principle **19**

3.1 The principle 19

3.2 Polynomial Size refutation 20

3.3 Degree lower bound for Graph ordering principle 21

4 Non Automatizability of Polynomial Calculus **27**

4.1 Proof Strategy 27

4.2 The tautology $F(C, w, r)$ 28

4.2.1 Polynomial encoding of $F(C, w, r)$ 30

4.3 Degree Lower bounds for $F(C, w, r)$ 32

4.4 Non automatizability of HN, Pc and Pcr 35

5 Pcr_k proof system **39**

5.1 Pcr_k definition 40

5.1.1 Complexity measures for Pcr_k 40

5.2 Relations between Pcr_k and other proof systems 41

5.3 Switching lemma for k-DNF and k-monomials 42
5.3.1 Decision trees for boolean functions 42
5.3.2 Random restriction 43
5.3.3 An application: lower bound in $\textsc{Pcr}_k$ for weak pigeonhole principle 44
5.3.4 A framework 45
5.4 A separation between $\textsc{Pcr}_k$ and $\textsc{Pcr}_{k+1}$ 45
5.5 A separation between $\textsc{Pcr}_k$ and $\textsc{Res}_k$ 47

6 Random CNF are hard for $\textsc{Pcr}_k$ 51
6.1 Random 3-CNF, encodings and $\textsc{Pc}$ lower bounds 51
6.2 How to restrict $Ax = b$ preserving expansion 52
6.3 Normal forms 53
6.4 Random restriction 55
6.5 Main result 55

7 Open problems 57

A Notions from Commutative Algebra 63
A.1 Rings 63
A.2 Ideals, Ring Homomorphisms, Quotient rings 64
A.3 Quotient ring representations of boolean function 66

B Notions from Computational Complexity 67
B.1 Complexity classes 67
B.1.1 Reduction and **NP**-completeness 68
B.2 Combinatorial circuits 68
B.2.1 Complexity measures on circuit 69
B.2.2 Algebraic circuits 69
B.2.3 Boolean circuits 69
B.3 Parameterized Computational Complexity 70
B.4 Expanders 71

Chapter 1

Introduction

Theorem proving is the most important activity of mathematicians. At the beginning of the 20^{th} century it was though that every mathematical statement was deemed for a proof or for a counterexample. In the Second International Congress of Mathematics in Paris in 1900, David Hilbert gave is famous speech in which he listed what he though were the most important open problems in mathematics at that time. He claimed:

> ... that every definite mathematical problem must necessary be susceptible of an exact settlement, either in the form of an actual answer to the question asked, or by the proof of the impossibility of its solution...

It is out of doubts that the 2^{nd} and 10^{th} problems are perfect examples of this kind of optimism.

2^{nd} **problem** Prove that the axioms of arithmetic are consistent.

10^{th} **problem** Find an algorithm to determine whether a given polynomial Diophantine equation with integer coefficients has an integer solution.

Gödel's Incompleteness results crashed such optimism in 1930. Any finite method of proving theorems which is powerful enough is either incoherent or unable to prove all the true statements in the theory. It was clear that the computational nature of theorem proving was the cause of such incompleteness. The first accomplishment of recursion and computation theory was to prove impossibility of solving some natural computational problems. Non provability and non computability were shown to be different perspectives of a unifying theme.

With the raise of electronic computation it is clear that computational impossibility is not just a theoretical investigation but also an actual obstacle to solve problems. From the applicative point of view there is no difference between a never ending computation and a computation too long to be ever completed. Efficiency is a major topic in computability, and the study computational complexity was started with this problem in mind (see [30] and [35] for seminal works in the area).

What about the relation between computations and proofs in this quantitative approach? What about the efficiency of theorem proving? Formal logic systems have been used to formalize mathematical reasoning and have been deployed in several applicative contexts like AI, optimization, planning and so on... then it is immediately obvious that the size of a proof and the time needed in finding it is a major concern.

Such problems were effectively framed in [26] and [27]. In the latter paper the field of propositional proof complexity was started, as the study of the efficiency of proving. One of the main problem arising in proof complexity is that of proving size lower bounds on the length of a proof for a particular tautology.

In this thesis we concentrate on a particular way of proving theorems. Hilbert himself proved that in some precise conditions a set of polynomials either has a common root (i.e. a solution) or there is a polynomial equation which witnesses the absence of common roots. Such result has been used as a tool to prove propositional theorems (actually to refute unsatisfiable propositions, which is equivalent). Several systems have been defined in this way, systems which have great importance for two reasons: they generalize the well-studied and widely applied Resolution proof system; well-known algorithms for polynomial systems solving, like the Gröbner Basis Algorithm, can be used for proving theorems.

We now give preliminary notions and definitions. We point to Appendix A for reference on basic algebra, and to Appendix B for reference on basic computational complexity.

1.1 Propositional Proof systems

Proof complexity is the study of how difficult it is to prove a mathematical statement. Such endeavour requires precise context and limits. The concept of "proof" is not primitive, it is a byproduct of the concepts of "verification process" and "acceptance". A proof is whatever makes an appropriate verification process to accept.

In this thesis we are interested in propositional statements and in deciding whether or not they are tautologies. The first general definition of a proof system has been given in [27].

Definition 1.1. TAUT *is the set of all propositional formulas which are true for any boolean assignment.*

A proof system for propositional logic (a ***propositional proof system****) is a polynomial time Turing machine M such that the image of the function computed by M is* TAUT*.*

Let ϕ be a propositional formula, we say ϕ has a proof in the system M if there exists x such that $M(x) = \phi$. We say that x is a proof of ϕ.

The intended meaning of this process is that M does only output a tautology. Then by definition any input that makes M output ϕ is an actual proof of ϕ. Furthermore the function computed by M is surjective in TAUT, i.e. any tautology has at least one proof. This proof system is correct and complete.

It is very easy to show a proof system if we do not care about efficiency issues. A simple proof system could get a statement and a truth table in input: the system evaluates the statement for every possible input and verify (a) that the truth table values match (b) the evaluation is always *true*. If both conditions are satisfied then the system outputs the statement itself; in any other case the system outputs a canonical tautology. Such system is correct and complete, but it suffers of the exponential length of the proof with respect to the length of the statement. We are interested in proof systems for which any statement ϕ has a short proof compared to the length of ϕ.

Definition 1.2. *A proof system is* polynomially bounded *if for any propositional tautology ϕ, there is a proof x of ϕ such that $|x| = |\phi|^{O(1)}$.*

It is easy to see that a counterexample for a formula can be verified efficiently, so TAUT is in **coNP**. Furthermore a formula is a tautology if and only if its negation is unsatisfiable. Satisfiability is **NP**-complete, its complement is **coNP**-complete, and so it is TAUT. For more information about these fundamental concepts we suggest to check Appendix B. The existence of a bounded propositional proof system would imply that there is a short way to check either if a formula has a satisfying assignment or if there is no one. This would mean **coNP**=**NP**. The study of propositional proof complexity arose as a strategy to show that **coNP**$\neq$**NP**, by proving that for any proof system there is a tautology which requires exponential size proof.

The actual proof complexity routine is to show exponential lower bounds for stronger and stronger proof systems, for more and more general classes of tautologies, hoping that at some point we will find techniques for proving a lower bound for the general case. This hope is considered by most researchers in the field to be way too optimistic. It is convenient to define the notion of relative "strength" among proof systems.

Definition 1.3. *A proof system A $f(n)$-**simulates** a proof system B if there is an algorithm T with the following properties*

- *T runs in time $f(|x|)$ on any input x.*
- *for any propositional tautology ϕ and proof x such that $B(x) = \phi$, we have $A(T(x)) = \phi$.*

*We say a proof system A **polynomially simulates** or p-**simulates** a proof system B if it $n^{O(1)}$-simulates B.*

It would be a major step to define a proof system M which polynomially simulates any other proof system. Such M would be polynomially bounded if and only if **coNP**=**NP**. In [41] the possibility of such "strongest" proof system is studied in relation with open problems regarding exponential time computation.

Tools in proof complexity have also been used to analyze proof search techniques used in practice. In the context of propositional logic, a proof search technique is in fact a CNF solver. Such solvers are widely used in planning, AI techniques, etc... [49], notice that a computation trace of one such system can be verified efficiently for consistence. This means that any of such systems induces a proof system. Despite the generality of proof system definition the vast majority of widely used systems are limited versions of Resolution (Res) system. We are going to present Res later, together with several others proof system.

If the major problem in proof complexity is the existence of short proofs, another very relevant question is to understand if there is an efficient way to find a proof of length similar to the shortest one. Such procedure is called automatization and depends on the proof system. It could also be possible that for a simpler proof system it is easy to find proofs close to the best, even if such proof is big. The following definition was given for the first time in [21].

Definition 1.4. *We say a proof system is f-**automatizable** if there is an algorithm A such that for any proposition ϕ with shortest proof Π, $A(\phi)$ returns a proof Π' for ϕ with the guarantee that $|\Pi'| \leq f(|\Pi|)$. We says a proof system is **automatizable** if is $n^{O(1)}$-automatizable and **quasi-automatizable** if it is $2^{\log^{O(1)} n}$-automatizable.*

1.1.1 Resolution

Resolution (Res) is the most famous and studied proof system in literature. Almost every theorem proving procedure and SAT-solver is a weak variant of Res. Resolution is most often used as a refutational system. The main purpose is to show that a formula implies a contradiction. Then such formula is unsatisfiable.

Resolution system focus on CNFs. This is not a significant restriction because any boolean formula can be transformed in a CNF which is satisfiable if and only if the original one is.

Given a boolean CNF ϕ on boolean variables $x_1, \ldots, x_n$, any line in a resolution proof is a clause over these variables. At any step of the proof one of the following rules is applied (A and B are clauses, x is a literal).

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B} \qquad \textit{resolution}$$

$$\frac{A}{A \vee x} \qquad \textit{weakening}$$

Thus a resolution refutation is a sequence of clauses, each of them being either (a) one of the clauses of ϕ, (b) obtained by a resolution rule from two previous lines or (c) obtained by weakening rule from a previous line.

We say the sequence of deduction is a refutation of ϕ if it contains the empty clause. The system is *sound* in the sense that the empty clause can only be deduced if ϕ is unsatisfiable: any assignment that satisfies all premises of a weakening or resolution rule also satisfies the consequent. It follows that an assignment that satisfies ϕ, also satisfies all clauses in a resolution proof. The empty clause is unsatisfiable, then either ϕ is unsatisfiable or the empty clause is not deducible.

There is a variant pf Resolution called **Treelike Resolution**. A resolution proof is a valid treelike resolution when all clauses but the initial ones are premises of at most one deduction each. Then the proof has a tree-like structure.

The fact that all unsatisfiable CNFs have a refutation is trivial, it follows from brute force procedures for satisfiability, which in turn can be easily transformed in treelike resolution proof.

Notice that such procedure is very inefficient, and requires exponential time though it can be easily implemented in linear space. It is then very interesting to study what are the necessary and sufficient resources needed to refute a CNF. Despite the simplicity of resolution, there haven't been result in this sense until Tseitin's paper in 1968 [53], and later Haken's 1985 [34].

The breakthrough result of Haken was to prove that a natural propositional formulation of the pigeon hole principle requires a superpolynomial number of steps. This lower bound has been improved to truly exponential. Following such results, a whole series of improvements, simplifications and variations have been developed in the area. Most notables are [24, 13, 17], and for a general treatment see [48].

Complexity measures for $\textsc{Res}$**:** let Π be a Resolution proof,

$S(\Pi)$ The **size** of Π, it is the number of symbols required to write the proof Π.

$|\Pi|$ The **length** of Π is the number of clauses appearing in Π.

$w(\Pi)$ The **width** of the proof, which is the biggest number of literals contained in a clauses of Π.

The size and length of a resolution proof are polynomially related. For a formula ϕ we customarily define $S(\phi)$ and $w(\phi)$ as the smallest size and the smallest width achievable (not necessarily simultaneously) in a refutation of ϕ.

It is well known that several restricted versions of resolution (like tree-like resolution) are strictly weaker that resolution, in the sense that there are formulas for which short (i.e. polynomial size) resolution refutations exist, but there is no short refutation in the restricted system [16, 6]. This is unfortunate, because almost every prover used in applications is even less powerful than tree-like resolution. Not to mention that there could be systems in which there are a very short proofs but no way to find them. It has been shown that finding the shortest resolution proof is NP-hard [38]. Even a proof only polynomially bigger is out of reach, unless some serious complexity assumptions fail [8].

1.1.2 Polynomially Calculus

We introduce three deductive proof systems in which lines are algebraic formulas. We interpret an algebraic formula $p(\vec{x})$ to be satisfied by a 0-1 assignment $\vec{b}$ if $p(\vec{b}) = 0$. If an algebraic formula p evaluates to 0 for any 0-1 assignment then we says p is a tautology, and we denote it as $\emptyset \models p$ or $\models p$.

If any 0-1 assignment which is a common root of $q_1, \dots q_m$ is also a root of p then we say $q_1, \dots q_m \models p$. We do not care about non 0-1 assignments. For any formula p such that $p(\vec{b}) = 0$ we denote $\vec{b} \models p$. This notation is standard in logic, and we just extend it to algebraic representations of boolean functions.

The *Polynomial Calculus* (PC) is a refutational system, defined in [25], and based on the ring $\mathbb{F}[x_1, \dots, x_n]$ of polynomials. To restrict the polynomials to be evaluated only on $\{0, 1\}$, the system contains the axioms $x_i^2 - x_i$ for all $i \in [n]$. Moreover it has two rules. For any $\alpha, \beta \in \mathbb{F}$, p, q polynomials and variable x:

$$\frac{p \qquad q}{\alpha p + \beta q} \quad \textit{Scalar addition}$$
$$\frac{p}{xp} \quad \textit{Multiplication}$$

A PC proof of a polynomial g from a set of initial polynomials $F = \{f_1, \dots, f_m\}$ (denoted by $F \vdash g$) is a sequence of polynomials where each one is either an initial one, an axiom or is deduced applying a rule to earlier polynomials.

A PC refutation is a proof of the polynomial 1. The rational of this proof system is that any 0-1 assignment which is a common root for the initial polynomials is also a root for any deduced polynomial. Thus the deduction of 1 is a proof that no such common root exists. A proof of

$$F \models p \iff F \vdash p$$

is given in Theorem 5.2 of [23]. In the algebraic language it means that $F \vdash p$ if and only if p is in the ideal generated by F and by the axioms $x_i^2 - x_i$ for all $i \in [n]$.

Complexity measures for PC**:** let Π be a PC proof,

$S(\Pi)$ The **size** of Π is the number of monomials appearing in Π.

$|\Pi|$ The **length** of Π is the number of polynomials appearing in Π.

$\deg(\Pi)$ The **degree** of Π is the maximal degree of a polynomial in Π.

Polynomial Calculus with Resolution (PCR) [4] is a refutational system which extends PC to polynomials in the ring $\mathbb{F}[x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n]$, where $\bar{x}_1, \dots, \bar{x}_n$ are new formal variables. PCR includes the axioms and rules of PC plus a new set of axioms defined by $1 - x_i - \bar{x}_i$ for all $i \in [n]$ to force $\bar{x}$ variables to have the opposite values of x variables.

We extend to PCR the definitions of proof, refutation, degree, size and length given for PC. Observe that using the linear transformation $\bar{x} \mapsto 1 - x$, any PCR refutation can be converted into a PC refutation without increasing the degree. Notice that such transformation could increase the size exponentially. Moreover PCR efficiently simulates RES with refutations of degree equals to the width of the original RES proof [11].

Nullstellensatz (HN) Nullstellensatz is a static refutational proof system bases on the Hilbert's Nullstellensatz theorem. It states that for any given set of multivariate polynomials $p_1 \dots p_l \in \mathbb{F}[x_1, \dots, x_n]$ where $\mathbb{F}$ is a closed field exactly one of the following may happen:

- There is a common root for such polynomials.
- There are polynomials $h_1 \dots h_l$ such that $\sum_{i=1}^{l} h_i p_i = 1$

The two things cannot happen simultaneously, nevertheless Hilbert's Nullstellensatz shows that almost one of them occurs. By adding axioms $x_i^2 - x_i$ we enforce the common root to be a 0-1 solution.

In our framework we are often interested in $\mathbb{F}$ to be a non closed field, but this is not an issue in this case: because of 0-1 solutions, the polynomials have a common root in the algebraic closure of $\mathbb{F}$ if and only if they have one in $\mathbb{F}$. Then assuming there is no such solution then

$$\sum_{i=1}^{l} h_i p_i + \sum_{i=1}^{n} g_i (x_i^2 - x_i) = 1$$

for some h_is and g_is. It is possible to show that in this case the coefficients of h_is and g_is are in $\mathbb{F}$.

Complexity measures for HN: let $\Pi = \{h_1, \ldots, h_l\}$ be a HN proof,

$S(\Pi)$ The **size** of Π is the number of monomials appearing in the polynomials Π.

$\deg(\Pi)$ The **degree** of an HN proof is $\max_i\{\deg(h_i p_i)\}$.

1.1.3 RES_k, a resolution on k-DNF

The Resolution system was extended by Krajíček in [40] to a system, called k-DNF Resolution (RES_k), where the formula used in each line is k-DNFs instead of a clause.

Resolution on k-DNF (RES_k) [40] is a sound and complete refutational system which extends *Resolution* (RES) to k-DNFs. The rules are the following:

$$\frac{A}{A \vee l} \qquad \textit{Weakening}$$

$$\frac{A \vee l_1 \quad \cdots \quad A \vee l_j}{A \vee \bigwedge_{i=1}^{j} l_i} \qquad \wedge\textit{-intro, } 1 < j \leq k$$

$$\frac{A \vee \bigwedge_{i=1}^{j} l_i}{A \vee l_i} \qquad \wedge\textit{-elim, } 1 < j \leq k$$

$$\frac{A \vee \bigwedge_{i=1}^{j} l_i \quad B \vee \bigvee_{i=1}^{j} \neg l_i}{A \vee B} \qquad \textit{Cut, } 1 < j \leq k$$

Complexity measures for RES_k: let Π be a Resolution proof,

$S(\Pi)$ The **size** of Π, it is the number of symbols required to write the proof Π.

$|\Pi|$ The **length** of Π is the number of k-DNFs appearing in Π.

1.2 Algebraic Representation of boolean functions

In this thesis we often represent boolean functions using algebraic formulas and circuits. On such representation we define complexity measures. We need to make clear how the language of boolean functions relates with the language of algebraic formulas.

Definition 1.5. *We denote the **boolean domain** as $\{0,1\}^n$ for a fixed n. We call **boolean function** any function from a boolean domain to an arbitrary image set.*

Given a field $\mathbb{F}$, we consider polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ as algebraic representations for functions with domain $\{0,1\}^n$ and range $\mathbb{F}$.

Definition 1.6. *We say that $p \in \mathbb{F}[x_1, \ldots, x_n]$* ***computes*** *a boolean function F if for all $\vec{v} \in \{0,1\}^n$ we have $F(\vec{v}) = p(\vec{v})$.*

Several polynomials represent the same boolean function. Thus we need a “canonical” representation.

Definition 1.7. *A* ***multilinear*** *polynomial is a polynomial in which no variable is raised to a power greater than one.*

Fact 1.8. *Any boolean function of n variables with values in a field $\mathbb{F}$ can be computed by a unique multilinear polynomial in $\mathbb{F}[x_1, \ldots, x_n]$.*

Proof. The proof of existence goes by induction on the number of variables. For constant functions it is trivial. Assume that the statement holds for $n-1$: any function F on n variables induces two functions on $n-1$ variables $F_0 := F\restriction_{x_n=0}$ and $F_1 := F\restriction_{x_n=1}$. By inductive hypothesis they are computable by two multilinear polynomials p_0 and p_1 without variable x_n. Then $x_n p_1 - x_n p_0 + p_0$ is a multilinear polynomial which computes F.

We now want to prove uniqueness. We start by proving that the zero function has exactly one multilinear representations: the empty polynomial. Given a polynomial we call “non-trivial” any monomial occurring with non zero coefficients.

We consider now a multilinear polynomial with some non-trivial monomials. We pick among them an arbitrary minimal monomial t with respect to inclusion (i.e. division) partial order. Consider the following assignment: the variables appearing in t are set to 1, the other ones are set to 0. Under such assignment all non-trivial monomials but t evaluate to 0, while t evaluates to some non-zero value. Thus the zero function can't be computed by a non empty polynomial.

For a general function let us assume that two different polynomials compute it. Their difference is not empty but computes the zero function, and this is a contradiction. $\square$

For a boolean function f to a field $\mathbb{F}$ we say $\deg(f)$ is the degree of its multilinear representation.

1.2.1 Symbolic variables for negation

In propositional logic it is customary to use the negation operator in propositional expression. In the algebraic setting the negation of an expression e which evaluates in $\{0,1\}$ is representable as $1-e$. As we will see later, this is an unfortunate choice if we want to represent CNF clauses efficiently with polynomials.

We manage this by using polynomials with $2n$ variable, usually denoted in the following as $x_1 \ldots x_n, \bar{x}^1 \ldots \bar{x}^n$. We may think about this polynomials as algebraic circuits with $2n$ input gates: n of them encode the regular input value, the other n encode its negation.

We then consider only $\{0,1\}$ assignments such that for any i the equation $\bar{x}_i = 1 - x_i$ holds. In such framework there are $\{0,1\}^n$ possible assignments, and any such assignment is specified by the values of $x_1 \ldots x_n$. Thus we can use algebraic expression over $2n$ variables (with negated variables) as representation of boolean functions.

Definition 1.9. *We say that $p \in \mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$* ***computes*** *a boolean function F if for all $(v_1 \ldots v_n) \in \{0,1\}^n$ the equation $F(v_1, \ldots, v_n) = p(v_1, \ldots, v_n, 1-v_1, \ldots, 1-v_n,)$ holds.*

This definition is merely an extension of the one of polynomials in n variables, but notice that uniqueness of representations of boolean functions doesn't hold for multilinear polynomials in $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$. Indeed the simple boolean function $\neg x_1$ can be represented by both $1 - x_1$ ad $\bar{x}_1$. This is the reason we introduced negated variables: among multiple representations there could be a small one.

1.3 Partial assignments and restrictions

Partial assignments to boolean functions and expressions are common in this thesis. We want to describe assignments in a way which is sensible for functions, polynomials and arbitrary algebraic circuits. Notice that a partial assignment acts in different ways to different description of a boolean function. We will prove that such different behaviors are compatible with the underlying semantic.

Definition 1.10. *We call ρ an* ***assignment*** *(or equivalently a* ***restriction****) any set of couples (x, b) where x is a variable and b is a value in $\{0, 1\}$ such that $(x, b) \in \rho$ implies $(x, 1 - b) \notin \rho$.*

We say x is ***assigned*** *to b by ρ if (x, b) is contained in ρ, and we name $\mathrm{Dom}(\rho)$ the set of variables assigned by ρ. For any x in $\mathrm{Dom}(\rho)$ we denote $\rho(x)$ to be the value x is assigned to by ρ. We denote as $\rho^{-1}(b)$ the set of variables assigned to b.*

Given two assignments ρ, ρ' we say they are ***compatible*** *if for any variable x in $\mathrm{Dom}(\rho) \cap \mathrm{Dom}(\rho')$ we get $\rho(x) = \rho'(x)$.*

For a family of pairwise compatible assignments we call the intersection of them the ***intersection assignment****; and we call the union of them the* ***union assignment****.*

The natural interpretation of a partial assignment ρ to act on a function F is to obtain a function F' where some of the parameters are fixed.

Definition 1.11. *Let be F a function and ρ an assignment. Consider a boolean function F' defined as follows: for any $\vec{b} \in \{0, 1\}^n$ the value of F' is equal to $F(b'_1, b'_2, \ldots, b'_n)$ where*

$$b'_i = \begin{cases} b_i & \text{when } x_i \notin \mathrm{Dom}(\rho) \\ \rho(x_i) & \text{when } x_i \in \mathrm{Dom}(\rho) \end{cases}$$

we call F' the ***restriction*** *of F by ρ, and we denote it as $F\restriction_\rho$.*

A partial assignment transforms a function by fixing some parameters. In a similar fashion we can think of a circuit in which some inputs are "hardwired". Assume an arbitrary combinatorial circuit C (see Section B.2) defined over a set of variables, and a partial assignment ρ, we consider the following process on C:

1. For any input gate labelled by a function f, the label is changed to be $f\restriction_\rho$.
2. If there is any internal gates for which some of the predecessors have been changed in the process, then the label of such gate is restricted accordingly. More precisely its label function is a new function for which the parameters corresponding to the constant predecessors are restricted to the corresponding values, and the edges are removed.
3. The process is closed with respect to the second step.
4. All gates that were not output gate before the process but have out degree 0 after the process are removed.
5. The process is closed with respect to the third step.

We defined restriction on both functions and circuits. As long as circuits describe functions it is worth to remark that for any circuit C which computes the function f and any partial assignment ρ, the circuit $C\restriction_\rho$ computes $f\restriction_\rho$.

1.4 Thesis's content

Chapter 2: Proof search techniques in Polynomial Calculus. We discuss proof search techniques in the context of algebraic proof systems. We introduce concepts from algebra and we show how they are related with proof search. We discuss the efficiency of such search methods and show it is deeply related with the required degree of a proof. We state a relation between size and degree required by a refutation in Polynomial Calculus. All results shown in this chapter are known in literature.

Chapter 3: Graph Ordering Principle. We show that there is a proposition with a short refutation and which requires high degree. This show that any proof search technique based only on degree is inefficient. This include Gröbner bases techniques presented in Chapter 2. Results in this chapter answer a question open by Bonet and Galesi in [19, 20] regarding the optimality of the relation between degree and size of proof in PC. This chapter comes from the paper

> [31] N. Galesi, M. Lauria. *Degree Lower bounds for a Graph Ordering Principle*, 2008 (Submitted manuscript)

Chapter 4: Non Automatizability of Polynomial Calculus. We prove that no automatization exists for algebraic proof systems HN, PC and PCR. We rely on work done by Alehknovich and Razborov [8] for Resolution proof system. Their result follows from a construction in two steps: (1) automatization of Resolution proof system implies approximation of an hard optimization problem; (2) a gap amplification technique gives an exact algorithm from the approximation. First step only depends on the actual proof system so we adapt that step for our proof systems and we obtain the result. This chapter comes from paper

> [32] N. Galesi, M. Lauria. *On Automatizability of Algebraic Proof Systems*, 2008 (Manuscript)

Chapter 5: PCR_k proof system. We study a proof system PCR_k which is the mixture of PCand RES_k. We show that for each constant value k PCR_k is exponentially stronger that PCR_{k-1} on some tautologies. We prove PCR_k is exponentially stronger that RES_k for some special forms of matching principle. We also sketch a proof for an exponential lower bound on a weak form pigeon hole principle. This chapter comes from paper

> [33] N. Galesi, M. Lauria. *Extending Polynomial Calculus with k-DNF Resolution*, 2007. Electronic Colloquium con Computational Complexity (TR07-041)

Chapter 6: Random CNF are hard for PCR_k. We study how the PCR_k proof system behaves with respect to random 3-CNF. We adapt a restriction presented in [2] to PCR_k proofs and we show that with high probability PCR_k requires exponentially big refutations for a randomly generated 3-CNF. This chapter comes from paper

> [33] N. Galesi, M. Lauria. *Extending Polynomial Calculus with k-DNF Resolution*, 2007. Electronic Colloquium con Computational Complexity (TR07-041)

Chapter 2

Proof search techniques in Polynomial Calculus

How to find proofs? In this chapter we discuss a general technique for proving theorems in propositional proof complexity by using polynomial calculus. We need concepts from basic commutative algebra and ring theory. Such concepts are non obvious for computer scientists so we give a brief reference about this topics in the Appendix A. This chapter doesn't contain new results, but it is necessary to put in context the whole algebraic proof complexity research area and to give a further justification for the results proved in Chapter 3.

Degree is an indirect complexity measure with a lot of merits. In the next chapters several analysis of this measure will lead to size lower bounds.

Here we want to show a proof search strategy for polynomial calculus whose performance is heavily influenced by the degree required by the refutation.

We have seen that a polynomial calculus refutation consists in proving that a system S of polynomial equations implies $1 = 0$. In particular we need to prove that 1 can be written as a algebraic combination of the polynomials in S. This is equivalent (see A) to show that 1 is contained in the ideal generated by the polynomials in S.

We give a primitive proof strategy to prove the completeness of Pc. Such strategy has exponential size in the number of variables involved in the system.

Trivial proof strategy: Consider incompatible polynomials $p_1 \dots p_l$. For any boolean assignment α there is a p_i such that $p_i(\alpha) = 1$. Consider the multilinear characteristic function χ_α for α. Then $p_i \vdash \chi_\alpha \cdot p_i \vdash \chi_\alpha$. We deduce $\sum_\alpha \chi_\alpha$ which is 1.

2.1 Order of monomials in residue

In case of multivariate polynomials it is non obvious how to define canonical elements for residue classes (see Definition A.12 in Appendix A). Consider for example the division of xy by $x - y$. By simple symmetry we get that both $xy = y^2$ and $xy = x^2$ are true modulo $x - y$, and no choice among them can be done by only relying on the algebraic structure. For a satisfactory definition of residue modulo an ideal we need an order among polynomials. This requirement is inherent to the algebraic structure of the polynomial ring, and it is needed to force the notion of a minimum element in an equivalency class. Then residue in multivariate polynomials is defined by giving a preferred order among monomials. More information about polynomial orders is in [28].

Graded lexical order: ($<_{\mathbb{P}}$) Consider variables $x_1, \dots, x_n$. Ordering among monomials is completely specified by the following properties (here m and m' are general monomials):

- $x_n <_{\mathbb{P}} x_{n-1} <_{\mathbb{P}} x_{n-2} <_{\mathbb{P}} \cdots <_{\mathbb{P}} x_2 <_{\mathbb{P}} x_1$

- If degree of m is smaller than degree of m' then $m <_{\mathbb{P}} m'$.
- If degrees of m and m' are equal, the lexicographic order applies.
- 1 is the smallest monomial.

Once a total order among monomials is given, we can extend such order to polynomials. For a polynomial $\alpha m + q$ such that m is the bigger monomial for which α is not zero, we call m the **leading monomial** and αm the **leading term**. Consider polynomials $p = \alpha m + q$ and $p' = \beta m' + q'$ where m and m' are the respective leading monomials.

- $m <_{\mathbb{P}} m'$ implies $p <_{\mathbb{P}} p'$
- $m = m'$ and $q <_{\mathbb{P}} q'$ imply $p <_{\mathbb{P}} p'$
- 0 is the smallest polynomial. (Notice that $0 <_{\mathbb{P}} 1$)

We integrate here the Definition A.12 given in Appendix A with the specification of the canonical element which represent each residue class.

Definition 2.1. *The residue of a polynomial p modulo the polynomial ideal I is the smallest r (according to grlex order) such that $f \equiv_I r$. We denote it as $R_I(f)$.*

Notice that $<_{\mathbb{P}}$ is not a total order: p and γp are incomparable for any $\gamma \notin \{0, 1\}$. This is not an issue for the definition of residue: if $f \equiv_I r$ and $f \equiv_I \gamma r$ with $\gamma \neq 1$ then we also have that $\gamma f - f \equiv_I \gamma r - \gamma r = 0$ so $f \equiv_I 0$. Then the residue is always a well defined minimum. Here we give two simple properties about the residue operator in the polynomial ring.

Fact 2.2. *Let I be a polynomial ideal and let p and q be two polynomials. Then:*

- $R_I(p) \leq_{\mathbb{P}} p$;
- $R_I(pq) = R_I(p \cdot R_I(q))$.

2.2 Division algorithm

A correct algorithm of *ideal membership* easily follows from the division algorithm for multivariate polynomials.

In this section we show how the division algorithm on polynomials is in fact a very efficient Pc proof search strategy, both in term of degree and size. Consider the Algorithm 1 (page 13) where we denote $LT(p)$ as the leading term of a polynomial p.

The value of r at the end is the **remainder** of f divided by $p_1, \ldots, p_l$. This algorithm has several nice properties: it gives a Pc proof of $f - r$ from polynomials $p_1 \ldots p_n$. The proof has polynomial size with respect to $p_1, \ldots, p_l$ and $f - r$. The degree of the proof is always below the degree of f.

There are two problems arising in this proof search strategy: the following example shows that the outcome is undefined unless we specify the order in which the polynomials $p_1 \ldots p_l$ are checked for divisibility.

Division of $x^2y + xy^2 + y^2$ by $\overbrace{(xy - 1, y^2 - 1)}^{\text{order matters}}$

$$x^2y + xy^2 + y^2 = (x + y) \cdot (xy - 1) + 1 \cdot (y^2 - 1) + \underbrace{x + y + 1}_{\text{remainder}}$$

Algorithm 1 Division

Input f, $p_1, \ldots, p_l$

Output $a_1, \ldots, a_l$, r such that $f = a_1p_1 + \ldots + a_lp_l + r$

```
1: procedure Division
2:     $g \leftarrow f$
3:     while $g \neq 0$ do
4:         for $i := 1 \ldots l$ do
5:             if $LT(p_i)$ divides $LT(g)$ then
6:                 $a_i \leftarrow a_i + \frac{LT(g)}{LT(p_i)}$
7:                 $g \leftarrow g - \frac{LT(g)}{LT(p_i)} \cdot p_i$
8:                 Break
9:         if $i > l$ then
10:            $r \leftarrow r + LT(g)$ , $g \leftarrow g - LT(g)$
```

Division of $x^2y + xy^2 + y^2$ by $(y^2 - 1, xy - 1)$

$$x^2y + xy^2 + y^2 = (x+1) \cdot (y^2 - 1) + x \cdot (xy - 1) + \underbrace{2x + 1}_{\text{remainder}}$$

The example implies that different orders in the list of divisors lead to different results. In particular we have $x - y = (2x + 1) - (x + y + 1)$, and $x - y$ cannot be divided by neither $xy - 1$ nor $y^2 - 1$. Thus the division algorithm can't prove that $x - y$ is in the ideal generated by $\{xy - 1, y^2 - 1\}$.

Theorem 2.3. *Given a set of polynomials $p_1, \ldots p_l$ and a polynomial f:*

- *If f divided by $p_1, \ldots p_l$ has remainder 0 then $p_1, \ldots, p_l \vdash f$.*
- *The minimal size of such proof is polynomial in the size of p_is and f.*
- *The minimal degree of such proof is less or equal than the maximum degree among p_is and f.*

2.3 Gröbner basis

The division algorithm does not produce a complete proof system. In this section we extend the division algorithm to design a complete proof search technique. Notice that such strategy is well known in algebraic geometry and it has been applied to Pc already in [25].

Given an ideal I we would like to find a set $G = \{g_1, \ldots, g_t\} \subseteq I$ such that

$$f \text{ divided by some order of } G \text{ has remainder } 0 \iff f \equiv_I 0$$

This would give a way to prove ideal membership for any element of I. Such sets G are called **Gröbner basis** of I (here we will call then just **basis**). We suggest to check [28] for more information about them. At the very least a base is a generator for I. Our main interest in such basis is because of the following easy facts:

Theorem 2.4. *Consider and ideal I and one of its Gröbner basis, G.*

- *the division algorithm has the same result whatever ordering is considered for the elements is G.*

- *for any f we get that f divided by G gives remainder $R_I(f)$.*

Proof. Consider two different remainders r_1 and r_2 for two running of the division algorithm with two different orders of G. Both r_1 and r_2 are not reducible by G, otherwise the division algorithm wouldn't have stopped. But $r_1 - r_2$ is a non zero polynomial in I. G is a base so we know $r_1 - r_2$ divided by some ordering of G has zero remainder. Then there is a monomial in $r_1 - r_2$ which is reducible by G, which implies one of the two initial running shouldn't have stopped. This is a contradiction, and r_1 is equal to r_2. We denote this polynomial as r.

We now know that any running of f divided by G gives the remainder r. To check if $r = R_I(f)$ we notice that neither $R_I(f)$ nor r have monomials which are divisible by a member of G, otherwise they would fail to be minimal. We also know that $r - R_I(f)$ divided by G must have remainder 0 because G is a base. It follows that r is equal to $R_I(G)$. □

The corollary to the properties shown above is that we can design an algorithm which check if a propositional formula is a theorem.

Given a propositional claim C

1. Transform C in a DNF preserving validity.
2. Take the dual CNF (De Morgan's laws).
3. Transform encode the CNF in a set of polynomials F.
4. Deduce a Gröbner base G from F (we will see how).
5. Divide 1 by G and check if remainder is 0 (i.e. check $1 \in G$).

All we need is to specify the process for the deduction of a Gröbner base from the ideal generators.

2.3.1 Computation of a Gröbner base

Before describing the actual computation we want to comment the counterexample we gave for the division algorithm. We saw that $xy - 1, y^2 - 1 \vdash x - y$, but neither $xy - 1$ nor $y^2 - 1$ divides $x - y$. But we know that the polynomial $x - y$ is an algebraic combination of the premises because of the following computation

$$x - y = (y) \cdot (xy - 1) - (x) \cdot (y^2 - 1)$$

Notice that a division can't produce this algebraic combination because it requires degree 3 polynomials, while target and premises have degree at most 2. The key fact is that $x - y$ has no monomial divisible by the leading term of a generator, and the algebraic combination of $xy - 1$ and $y^2 - 1$ cancels those leading terms and highlights lower terms. Cancellation among leading terms does the trick not only in this case, it is sufficient to obtain all elements of a Gröbner base. We will introduce the Buchberger algorithm for basis computation. The main concept of this algorithm is that of S-polynomials, which formalize cancellation of the leading terms.

Definition 2.5. *Let $p_1 = \alpha m_1 + q_2$ and $p_2 = \beta m_2 + q_2$ two polynomials such that m_1 and m_2 are the respective leading monomials. Consider m the least common multiple of m_1 and m_2. We define*

$$S(p_1, p_2) := \frac{m}{\alpha m_1} \cdot p_1 - \frac{m}{\beta m_2} \cdot p_2 = \frac{m}{\alpha m_1} q_1 - \frac{m}{\beta m_2} q_2$$

It is called the S-polynomial of p_1 and p_2

Algorithm 2 Buchberger's Gröber base computation

Input $F = \{p_1, \dots, p_l\}$

Output G a Gröbner base for the ideal I

```
1: procedure BUCHBERGER($F$)
2:     $G \leftarrow F$
3:     repeat
4:         $G' \leftarrow G$
5:         for $p, q \in G', p \neq q$ do
6:             $s \leftarrow$ remainder of $S(p,q)$ divided by $G'$
7:             $G \leftarrow G \cup \{s\}$
8:     until $G \neq G'$
```

With S-polynomials in hand we finally give the Algorithm 2 (page 15) for Gröbner base computation.

For the proof of completeness of Algorithm 2 we direct the reader to standard literature [28]: the main idea is that by the means of S-polynomials we enrich the set of leading monomials in G. The algorithm stops when for any leading monomial m of a polynomial in I there is a polynomial in G whose leading monomial is a divisor of m. If G satisfies such property then the division algorithm does a nontrivial step for any polynomial in I. By induction it is easy to prove that a polynomial in I has remainder zero with respect to G.

Theorem 2.6. *Given a set of polynomials F, Algorithms 2 computes a Gröbner base of the ideal generated by F.*

The algorithm deserves some comments. (a) The boolean axioms are in F, so at any step the set $G-F$ contains only multilinear axioms because the division against the boolean axioms removes all squares. (b) There is at most one polynomial in $G-F$ for any possible leading monomial. Also no leading monomial occurs in a non leading position in $G-F$. This means that in a Gröbner base there are $O(n^d)$ elements of degree d, each of them has size $O(n^d)$. This observation is important to study the size of a Pc refutation by the means of the required degree.

2.4 Bounded degree Gröbner base computation

Degree matters when studying Pc proofs. The strategy discussed so far gives a refutation in Pc, but there is no nontrivial upper bound on the size of the base computed.

Notice that the Algorithm 2 is not deterministic in the sense that at any step p and q are chosen arbitrarily. This leads to different basis from the same set of generators. It would be nice to deduce a small one.

Algorithm 2 does not give any guarantee about this requirement: consider two set of polynomial equations over disjoint variables with the following properties: (a) they are both unsatisfiable (b) the first one requires a big base (c) the second one has a small base. In this case there are choices leading to a small base and others leading to a big one.

How difficult is to find a small Pc proof for propositional theorem? This problem will be discussed in its full generality in Chapter 4. Here we give a first answer based on the degree required by the proof. If we can keep the algorithm below a certain degree d then we can guarantee the number of elements in the base is $n^{O(d)}$. An approach is that of giving priority to S-polynomials which are deducible in degree less that d. Eventually the algorithm either finds a base or manages the failure, for example by increasing the said degree threshold or claiming there is no proof of degree less d.

It would be nice if a bounded degree version of the Buchberger algorithm could characterize all ideal members provable in a fixed degree. Then as in the general case the algorithm would be complete for bounded degree Pc proofs.

This is indeed true and the bounded version of the algorithm is very similar to the unbounded one. Operation used in Algorithm 2 are divisions and S-polynomials. Divisions never increase the degree, while the Pc deduction of an S-polynomial may require intermediate steps of degree higher than the ones of previous steps. To enforce degree below d we need to modify the step of the S-polynomial computation.

Algorithm 3 Bounded degree Buchberger's Gröber base computation

Input $F = \{p_1, \dots, p_l\}$, d.

Output A set of d degree polynomials in the ideal I

```
1: procedure Buchberger($F$,$d$)
2:     $G \leftarrow F$
3:     repeat
4:         $G' \leftarrow G$
5:         for $p, q \in G', p \neq q$ do
6:             if $S(p,q)$ is not deducible for $p, q$ in degree $d$ then
7:                 continue
8:             $s \leftarrow$ remainder of $S(p,q)$ divided by $G'$
9:             $G \leftarrow G \cup \{s\}$
10:    until $G \neq G'$
```

The algorithm describes a Pc proof of degree at most d. Actually it gives a the complete proof strategy we are looking for. This statement is not proved here.

Theorem 2.7. *Given a set of polynomials F and a positive integer d, Algorithm 3 outputs a set G_d such that for every p provable from F in degree at most d, p divided by G_d (in any order) gives remainder 0.*

To have a degree d proof (or to check it does not exist) is sufficient to follow the next procedure:

1. Transform C in a DNF preserving validity (standard).
2. Take the dual CNF (De Morgan's laws).
3. Encode the CNF in a set of polynomials F.
4. $G_0 = F$
5. Compute G_1 :=Buchberger$(G_0, 1)$
6. If $1 \in G_1$ accept.
7. Compute G_2 :=Buchberger$(G_1, 2)$
8. If $1 \in G_2$ accept.
9. Compute G_3 :=Buchberger$(G_2, 3)$

10. ...

11. Compute G_d :=BUCHBERGER(G_{d-1}, d)

12. If $1 \in G_d$ accept.

This procedure outputs a base of lowest degree possible, because as soon as there is a degree d' refutation, then $G_{d'}$ contains 1 by the Theorem 2.7.

Corollary 2.8. *Given propositional formula with a* PC *proof of constant degree, then Algorithm 3 outputs a proof in polynomial time.*

Corollary 2.9. *There exists a formula such that any Resolution proof has exponential size, and Algorithm 3 outputs a* PC *proof in polynomial time.*

Proof. **(sketch)** Consider a 3-regular bipartite expander graph with respectively n and $n+1$ vertices on the two sides. The principle stating that there is no matching between the two sets of vertices can be proved in PC using constant degree, and then Algorithm 3 accepts in polynomial time.

Any RES proof for this principle requires linear width for Theorem B.10 in Appendix B. Theorem 2.10 (later in this chapter) implies RES requires exponential size proof. □

2.5 Degree complexity of a proof

The running time of the Algorithm 3 heavily depends on the degree it considers before stopping. Then it is important to study degree complexity of algebraic proof. Several questions arise regarding the Algorithm 3:

- Does it exist a propositional theorem whose proof in PC requires high degree?
- Does it exist a theorem for which exists a short proof but the required degree is high?
- Does it exist a theorem for which any short proof has high degree but there are low degree proofs?

The first question has a well known positive answer. The works in [15, 7] give several examples of propositional theorems which require linear degree in the number of variables. In Chapter 3 we answer positively the second question by showing a propositional theorem on n variables with polynomial size proof and required degree $\sqrt{n}$.

Regarding the third question, the negative answer is given in [25, 17, 15, 37, 3] for Resolution and Polynomial Calculus. In this works the width and degree measures are related with the size measure. It turns out that we can reduce the degree of a PC proof to some extent by increasing its size:

Theorem 2.10. *Let F be a set of polynomials on n variables and degree at most d. If there is a* PC *or* PCR *refutation of F of size S then there is a refutation of F of degree at most D where*

$$D \leq O(\sqrt{n \log S}) + d$$

The previous statement tells us that we can obtain a low degree proof when the size S is not too big. Turning the argument around we get an important corollary: proving degree lower bound for PC immediately gives size lower bound for both PC and PCR. This is because degree complexity is the same in both proof systems.

Corollary 2.11. *Let F be a set of polynomials on n variables and degree at most d. If* Pc *refutations require degree at least D then any* Pc *or* Pcr *refutation has size at least $2^{\Omega(\frac{(D-d)^2}{n})}$.*

Proof. (Theorem 2.10, adapted from [3]) We prove the claim for a Pcr refutation size S. Fix $D = \sqrt{n \log S}$. Let S^* the number of monomials of degree greater than D in such proof, and E the smallest integer such that $S^* < \left(1 - \frac{D}{2n}\right)^{-E}$. We proceed by double induction on E and n to prove that there exists a refutation of degree $D + E + d$. If $E = 0$ then $S^* = 0$ so the $D + d$ degree proof exists. Assuming $E > 0$ we pick x to be the variable present in the biggest number of monomials. By counting we know x appears in at least $\frac{D}{2n}S^*$ monomials: restricting variable x to 0 (and its corresponding negation to 1) gives a proof of $F\restriction_{x=0}$ with at most $\left(1 - \frac{D}{2n}\right)^{-E+1}$ monomials of degree above D. Then by induction $F\restriction_{x=0} \vdash 1$ in degree at most $D + E - 1 + d$. For any $p + xq \in F$ we get $\bar{x}(p + xq) = \bar{x}p + \bar{x}xq \vdash \bar{x}p = \bar{x}((p + xq)\restriction_{x=0})$. We can conclude $F \vdash \bar{x} \cdot F\restriction_{x=0} \vdash \bar{x}$ in degree $D+E+d$. It is easy to check that $\bar{x}, F \vdash F\restriction_{x=1}$ in degree $d+1$. By inductive hypothesis $F\restriction_{x=1} \vdash 1$ in degree at most $E + D + d$. Concluding: $F \vdash F, \bar{x}F\restriction_{x=0} \vdash F, \bar{x} \vdash F\restriction_{x=1} \vdash 1$. The proof described so far is a Pcr one. By substituting any $\bar{x}$ variables with $1 - x$ we obtain a Pc refutation of the same degree. □

This Corollary 2.11 tells us that high degree complexity does limit *any* proof search strategy, not only the ones determined by the Buchberger algorithms. From the statement we also notice that degree complexity implies the same size lower bound for both Pc and Pcr.

2.5.1 Scheme of degree lower bounds

In Chapter 3 and 4 we will prove two lower bounds on the degree required by two specific refutations. Here we give an outline of the proof strategy.

Consider a set of polynomials F and an operator ψ such that the following properties hold:

$$\begin{aligned} \psi(p) &= 0 && \text{for each } p \in F \\ \psi(p+q) &= \psi(p) + \psi(q) \\ \psi(x \cdot p) &= \psi(x \cdot \psi(p)) \end{aligned}$$

Notice that for any Pc deduction, if ψ maps the premises to zero then it also maps to zero the consequence. Thus everything polynomial deducible from F is in the kernel of ψ. This means either $\psi(1) = 0$ or F is not refutable (i.e. satisfiable), and the two events cannot happen simultaneously.

Following the same intuition we consider a "pseudo-homomorphism", which is indistinguishable from an actual homomorphism by any refutation of degree below a certain threshold d. More precisely we want ϕ such that

$$\begin{aligned} \phi(p) &= 0 && \text{for each } p \in F \\ \phi(p+q) &= \phi(p) + \phi(q) \\ \phi(x \cdot p) &= \phi(x \cdot \phi(p)) && \text{if } p \text{ has degree less than } d \end{aligned}$$

It is clear that any polynomial which is deducible by Pc in degree less than or equal to d is in the kernel of ϕ. Thus if the initial set F has a refutation of degree at most d then $\phi(1) = 0$.

If we exhibit an operator which is a pseudo-homomorphism against deductions up to degree d, it maps F to 0 and does not map 1 to 0 then we get a lower bounds of degree d for any refutation of F.

Chapter 3

Graph Ordering Principle

Investigating the efficiency of the proof search algorithm proposed by Ben-Sasson and Wigderson [17], Bonet and Galesi [18, 19] proved that the class of formulas GT_n defined in [52] has polynomial size Resolution proofs but requires a $\Omega(n)$ (notice that GT_n uses n^2 variables). Thus they proved the optimality for the algorithm proposed in [17]. In [18, 19, 20] they tried to extend the previous results to PC. Informally they asked the following question: what is the efficiency of Algorithm 3 when compared to [17]? Can this algorithm perform better? They conjectured this is not the case in general, suggesting that some modifications of the GT_n principle would require linear degree refutations in Pc. They gave some partial results in this direction, showing that a modification of the Pigeon Hole Principle admits polynomial size Resolution refutations but requires $O(\log n)$ degree in Polynomial Calculus [20].

We now show a formula for which there exists a short Polynomial Calculus refutation but nevertheless this formula requires high degree to be proved. The required size and the required degree of this formula is tight for the size degree trade-off discussed in Theorem 2.10. This solves the conjecture of Bonet and Galesi [18, 19].

3.1 The principle

The *Graph Ordering Principle* [50] states that any graph with any total order on its vertices has at least a sink vertex: a vertex whose neighbors are later elements in the total order. This is a variant of the *linear ordering principle* (GT_n) defined in [52] and used in [19] for proving width-size tradeoff optimality.

For any two distinct numbers a, $b \in [n]$, consider a variable x_{ab} to be 1 is a is less than b in the total order, and 0 if a is greater than b.

We can easily express GT_n in resolution

$$\begin{array}{ll} \forall a < b & x_{ab} \vee x_{ba} \\ \forall a < b & \bar{x}_{ab} \vee \bar{x}_{ba} \\ \forall a, b, c & x_{ab} \wedge x_{bc} \implies x_{ac} \\ \forall a & \bigvee_{b \neq a} x_{ba} \end{array}$$

The x variables express the order relation. The first two conditions ensure this relation is antisymmetric, the third requires it to be transitive. The fourth condition requires that for any vertex there is a smaller element. This is clearly unsatisfiable, because any finite ordered set has a minimum.

This formula has short refutation in Polynomial Calculus (actually it can be easily refuted in Resolution). This fact has no consequences of size-degree tradeoff because the initial degree required to define the formula is equal to the number of vertices which is non constant in the number of variables in the formula itself.

To solve this problem we use Graph Ordering Principle on a graph G ($\textsc{Gop}(G)$). This is a modified version of GT_n appeared in [50] for similar purposes. $\textsc{Gop}(G)$ requires that any vertex has a smaller adjacent. This formula is unsatisfiable for any graph G because the least element in the order is a sink.

Furthermore the propositional formulation of $\textsc{Gop}(G)$ is given in a slightly different way than GT_n. The transitivity for three vertices is expressed by two clauses which exclude the possibility of any cyclic orientation.

$$\begin{array}{ll}
\forall u < v & x_{ab} \vee x_{ba} \\
\forall u < v & \bar{x}_{ab} \vee \bar{x}_{ba} \\
\forall u < v < w & x_{uv} \vee x_{vw} \vee x_{uw} \\
\forall u < v < w & x_{uv} \vee x_{vw} \vee x_{uw} \\
\forall u & \bigvee_{v \in \Gamma(v)} x_{vu}
\end{array}$$

Notice that this principle is expressible in width equal to the maximum degree of the underlying graph G. For any graph of n vertices, this principle subsumes GT_n (i.e. Gop(K_n)) by weakening. Thus it has a small refutation in term of the number of vertices. In the following we will give a degree lower bound of $\Omega(n)$, which is linear in the degree achieved by the known refutations.

We now encode the formula as polynomials. Notice that with the exception of boolean axioms all of them are monomials.

$$\begin{array}{ll}
\forall u, v & x_{uv}^2 - x_{uv} \\
\forall u < v & 1 - x_{uv} - x_{vu} \\
\forall u < v < w & x_{uw} \cdot x_{wv} \cdot x_{vu} \\
\forall u < v < w & x_{uv} \cdot x_{vw} \cdot x_{wu} \\
\forall u & \prod_{v \in \Gamma(u)} x_{uv}
\end{array}$$

Boolean axioms in the first two rows imply the relation is a well defined complete orientation. The third and fourth row state that a 3-cycle is a forbidden orientation. The last set of equations means that a sink in the graph is forbidden: for a vertex u we name M_u the corresponding polynomial. We name M_U the polynomials corresponding to a set of vertices U. We denote as $\mathcal{T}$ the set of all other polynomials.

3.2 Polynomial Size refutation

GT_n has a Resolution refutation of polynomial size [19]. This immediately implies there exists a $\textsc{Pc}$ refutation of $\textsc{Gop}(G)$ because (1) $\textsc{Gop}(G)$ subsumes GT_n, (2) $\textsc{Pcr}$ simulates Resolution, (3) the principle already contains opposite variables, so there is no difference between $\textsc{Pc}$ and $\textsc{Pcr}$ in this case. For completeness we now give a refutation of $\textsc{Gop}(G)$ in $\textsc{Pc}$. Such proof is an adaptation of the ones in [52, 19, 50], for our $\textsc{Pc}$ formulation.

Theorem 3.1. *Let G be an n vertices graph.* $\textsc{Gop}(G)$ *has a degree n* $\textsc{Pc}$ *refutation of size $O(n^4)$.*

Proof. We deduce $\mathrm{Gop}(K_n)$ from $\textsc{Gop}(G)$ by repeated application of product rule from $\prod_{v\in\Gamma(u)} x_{uv}$ to $\prod_{v\neq u} x_{uv}$ for any $u \in V(G)$. We now proceed by giving a polynomial size deduction of $\mathrm{Gop}(K_{n-1})$ from $\mathrm{Gop}(K_n)$. Fix an index $i < n$. Then for any $j \notin \{i, n\}$ we deduce $x_{jn}x_{i1}x_{i2}\cdots x_{i(n-1)}$ by applying the following scheme

$$\frac{\dfrac{\dfrac{x_{i1}x_{i2}\cdots x_{i(n-1)}x_{in} \qquad x_{ij}x_{jn}x_{ni}}{\vdots}}{x_{jn}x_{i1}x_{i2}\cdots x_{i(n-1)}x_{in}} \qquad x_{jn}x_{i1}x_{i2}\cdots x_{i(n-1)}x_{ni}}{\dfrac{\vdots}{x_{jn}x_{i1}x_{i2}\cdots x_{i(n-1)}}}$$

Denote $A_i = x_{i1}x_{i2}\cdots x_{i(n-1)}$. For any $j < n$ we just deduced $A_i x_{jn}$, which easily implies $A_i - A_i x_{nj}$ by boolean axioms. Also notice that $A_i - A_i t$ and $A_i - A_i t'$ imply $A_i - A_i tt'$. This observation allows to prove $A_i - A_i \prod_{j<n} x_{nj}$. By using the axiom $\prod_{j<n} x_{nj}$ we get A_i. We repeat for all i and we obtain $Gop(K_{n-1})$. In $n-3$ steps we deduce $Gop(K_3)$, which easily implies 1 in constant degree. This refutation needs $O(n^4)$ steps. □

3.3 Degree lower bound for Graph ordering principle

In this section we show that graph ordering principle requires the biggest degree possible for a formula with a polynomial size refutation. Ordering principles have been considered in [19] to prove the optimality of the size-width relation [17] for resolution. We do the same for polynomial calculus.

A simple example shows that the structure of the graph must be taken in consideration for the lower bound. Indeed consider G to be the path graph of n vertices $1, 2, \ldots n$. In this case $\textsc{Gop}(G)$ consists in polynomials

$$\begin{array}{ll} x_{n(n-1)} & \\ x_{12} & \\ x_{i(i-1)}x_{i(i+1)} & \qquad\qquad \forall 1 < i < n \end{array}$$

For any i we get that $\{x_{i(i+1)} \quad x_{(i+1)i}x_{i(i-1)}\} \vdash x_{i(i-1)}$. By induction we can deduce x_{21}. Also x_{12} is in the principle, thus 1 is deducible. This proof has degree 2, thus no nontrivial degree lower bound for $\textsc{Gop}(G)$ exists for general a graph G.

Theorem 3.2. *There exists a uniform infinite family $\mathcal{G}$ of simple graphs of constant degree such that for any G in $\mathcal{G}$ the principle* $\textsc{Gop}(G)$ *has polynomial size in $|V(G)|$ and any* $\textsc{Pc}$ *refutation of* $\textsc{Gop}(G)$ *requires degree $\Omega(|V(G)|)$.*

As we said previously we require specific conditions on the graph G to prove any lower bound for $\textsc{Gop}(G)$. In particular we need a graph with good expansion properties. Given such graph we exhibit a linear operator $\mathcal{L}$ which maps to 0 all low degree consequences of $\textsc{Gop}(G)$, and does not map 1 to 0. This means 1 is not derivable in low degree.

Lemma 3.3. *Let G be a (r, c)-vertex expander. There exists a linear operator $\mathcal{L}$ defined on polynomials such that*

1. *$\mathcal{L}(p) = 0$, for all polynomial $p \in$* $\textsc{Gop}(G)$

2. for each monomial t and variable x, if deg$(t) < cr/4$, then $\mathcal{L}(x \cdot t) = \mathcal{L}(x \cdot \mathcal{L}(t))$

3. $\mathcal{L}(1) = 1$.

We postpone the proof of this lemma to the end of the section. Now we show that Lemma 3.3 implies the following corollary.

Corollary 3.4. *If G is an (r, c)-vertex expander then there is no* Pc *refutation of* Gop(G) *of degree less than or equal to $cr/4$.*

Proof. Assume for the sake of contradiction that such refutation exists. Apply $\mathcal{L}$ on all its lines. Any premise or axiom in Gop(G) is set to 0 because of property (1) of $\mathcal{L}$ as stated by Lemma 3.3; any result of the sum rule is set to 0 if the premises are set to 0 because of linearity of $\mathcal{L}$; any application of the product rule of Pc keeps all terms below degree $cr/4$ by assumption, so property (2) of $\mathcal{L}$ implies that the result of such application is 0. By induction on the lines of the proof we get that the whole refutation is mapped to 0. This is a contradiction because the last line (i.e the polynomial 1) is not mapped to 0 according to property (3) of $\mathcal{L}$. □

In the following we assume G to be given and to be an (r, c)-vertex expander. All the definitions below are given with respect to such graph.

We now give an overview of the proof of Lemma 3.3. We need a linear operator which captures low degree reasoning, i.e. its kernel contains all polynomials deducible in low degree from Gop(G).

For any polynomial p in a refutation of Gop(G) we consider a vertex $u \in G$ such that M_u is a required in the proof of p. We may think of a polynomials which requires few vertices as a "locally deducible" polynomial.

We now focus on monomials. For a term t in the proof we isolate a set of relevant vertices (the *Support* in Definition 3.5) such that t depends on them. We prove that if G is a good vertex expander, then low degree derivable terms in a refutation of Gop(G) depend on supports of small size (Lemma 3.6). We also prove that for locally deducible terms the support is sufficient to deduce them (Lemma 3.7, 3.8, 3.9). Then any line in a low degree deduction can be rewritten as linear combination of polynomials, each of them deducible from a small support. We define the operator $\mathcal{L}$ in such a way its kernel contains the ideals generated by small supports so that anything deducible in low degree is in its kernel. Since by construction the kernel does not contain 1, we get the degree lower bound.

Definition 3.5. *We call $Vertex(p)$ the set of vertices which appear in the variables occurring in p. Given a set of vertices U we define the inference relation $\leadsto_U$ in this way: For $A, B \subseteq [n]$,*

$$A \leadsto_U B \quad \textit{if} \quad |B| \leq \frac{r}{2} \quad \textit{and} \quad \Gamma(B) \subseteq A \cup U$$

Sup(U), the support of U, is defined as the closure of $\emptyset$ with respect to $\leadsto_U$. We denote by Sup(p) the set Sup$(Vertex(p))$ for any polynomial p.

The notion of support is closely related with the notion of neighborhood in a graph: *Sup*(U) is the maximal set of vertices whose neighborhood is inside U and which is not big enough to break the expansion barrier r. The following lemma gives the link between the vertex expansion and degree of monomials: a small set of vertices (hence a low degree term) has small support.

Lemma 3.6. *If a set U has size less or equal than $cr/2$ then Sup(U) has size less or equal than $r/2$. If a monomial t has degree less than $cr/4$ then Sup(t) has size less or equal than $r/2$.*

Proof. Let $Sup(U) = I_1 \cup I_2 \cup I_3 \cup \dots \cup I_l$ where each I_i is the set added in the i-th step of the inference. Assume it has size strictly greater than $r/2$, then there is a step j where such size is overcome. Let us denote $A = I_1 \cup \ldots \cup I_{j-1}$ and $I = I_j$. Then $|A| \leq r/2$ and $|A \cup I| > r/2$. Also $|I| \leq r/2$ because of the size constraint in the definition of $\rightsquigarrow_U$. Then $|A \cup I| \leq r$ and hence by the vertex-expansion condition $|\Gamma(A \cup I)| > cr/2$. This proves the first part since $\Gamma(A \cup I) \subseteq U$.

The second part follows since the vertices appearing in term t are at most twice the degree of t. □

Lemma 3.7. *Let t be a term. For any not empty set of vertices A of size less or equal than $r/2$ and such that $A \cap Sup(t) = \emptyset$, there exists an edge $\{u, v\}$ in G such that $v \in A$, $u \notin Sup(t) \cup A \cup Vertex(t)$.*

Proof. By definition of $Sup(t)$ and the hypothesis of the lemma, it follows that $Sup(t) \not\rightsquigarrow_{Vertex(t)} A$. Then $\Gamma(A) \not\subseteq Sup(t) \cup Vertex(t)$, therefore there is an vertex in $\Gamma(A)/(Sup(t) \cup Vertex(t))$. □

A partial assignment ρ to the variables of $\text{GOP}(G)$ is a u-cta (critical truth assignment) when it sets u as a global minimum.

$$\rho = \begin{cases} x_{a,u} = 1 & \forall a, a < u \\ x_{u,a} = 0 & \forall a, u < a \end{cases}$$

Using this critical assignment we can show how the residue with respect to a small set of polynomials in $M_{[n]}$ is not smaller than the residue with respect to the support. Recall the definition of residue with respect to a polynomial ideal $R_I(p)$ in Chapter 2, Definition 2.1.

Lemma 3.8. *Let t be a term. Let I be a set of vertices such that $|I| \leq r/2$ and $I \supset Sup(t)$. Then there exists a $v \in I/Sup(t)$ such that:*

$$R_{\mathcal{T},M_I}(t) = R_{\mathcal{T},M_{I-\{v\}}}(t)$$

Proof. Applying Lemma 3.7 to t and $I/Sup(t)$ we get an edge $\{u, v\}$ such that $v \in I/Sup(t)$ and $u \notin I \cup Vertex(t)$. Let ρ be a u-cta. Note that any polynomial in $\mathcal{T}$ containing the vertex u is satisfied by ρ. Any other polynomial in $\mathcal{T}$ is not touched, so $\mathcal{T}\restriction_\rho \subseteq \mathcal{T}$. Moreover since $u \notin Vertex(t)$, $t\restriction_\rho = t$. Finally note that $M_I \restriction_\rho \subseteq M_{I-\{v\}}$ since ρ is setting to 0 at least M_v. Recall that if $A \vdash p$ and $B \supseteq A$ then $B \vdash p$. Thus we have the following derivations:

$$\begin{aligned} \mathcal{T}, M_I &\vdash t - R_{\mathcal{T},M_I}(t) && \text{By definition of } R && (3.1) \\ \mathcal{T}\restriction_\rho, M_I\restriction_\rho &\vdash t\restriction_\rho - R_{\mathcal{T},I}(t)\restriction_\rho && \text{By restriction from (3.1)} && (3.2) \\ \mathcal{T}, M_{I-\{v\}} &\vdash t - R_{\mathcal{T},M_I}(t)\restriction_\rho && \text{By previous observations on (3.2)} && (3.3) \end{aligned}$$

From (3.3) and minimality of the residue we then have that $R_{\mathcal{T},M_{I-\{v\}}}(t) \leq_{\mathbb{P}} R_{\mathcal{T},M_I}(t)\restriction_\rho$. Moreover, since $\mathcal{T}, M_I \vdash t - R_{\mathcal{T},M_{I-\{v\}}}(t)$, we have that $R_{\mathcal{T},M_I}(t) \leq_{\mathbb{P}} R_{\mathcal{T},M_{I-\{v\}}}(t)$, also by minimality. Finally $R_{\mathcal{T},M_I}(t)\restriction_\rho \leq_{\mathbb{P}} R_{\mathcal{T},M_I}(t)$ holds since a restriction can only decrease the order of a polynomial. Hence it must be $R_{\mathcal{T},M_{I-\{v\}}}(t) = R_{\mathcal{T},M_I}(t)$. □

Lemma 3.9. *Let t be a term. For any set of vertices I of size less than or equal to than $r/2$ and such that $I \supseteq Sup(t)$, the following holds:*

$$R_{\mathcal{T},M_I}(t) = R_{\mathcal{T},M_{Sup(t)}}(t)$$

Proof. If $I = Sup(t)$ then $R_{\mathcal{T},M_I}(t) = R_{\mathcal{T},M_{Sup(t)}}(t)$. If I is strictly bigger than S, then by lemma 3.8 there is a vertex $v \in I/Sup(t)$ such that $R_{\mathcal{T},M_I}(t) = R_{\mathcal{T},M_{I-\{v\}}}(t)$. The lemma follows by induction on the size of $I/Sup(t)$. □

Lemma 3.10. *For any term t, $Vertex(R_{\mathcal{T},M_{Sup(t)}}(t)) \subseteq Sup(t) \cup Vertex(t)$.*

Proof. Assume for the sake of contradiction that there is a node $u \in Vertex(R_{\mathcal{T},M_{Sup(t)}}(t))$ not in $Vertex(t) \cup Sup(t)$. Consider a u-cta ρ. By an argument analogous to that of Lemma 3.8 we then have $R_{\mathcal{T},M_{Sup(t)}}(t) \leq_{\mathbb{P}} R_{\mathcal{T},M_{Sup(t)}}(t)\restriction_{\rho} <_{\mathbb{P}} R_{\mathcal{T},M_{Sup(t)}}(t)$. □

We are ready to give the proof of Lemma 3.3.

Proof. **Lemma 3.3**
For any monomial t, the linear operator $\mathcal{L}(t)$ is defined by

$$\mathcal{L}(t) := R_{\mathcal{T},M_{Sup(t)}}(t)$$

and is extended by linearity to any polynomial.

First we prove that for any polynomial $p \in \textsc{Gop}(G)$, $\mathcal{L}(p) = 0$. If p is in $\mathcal{T}$, then $R_{\mathcal{T}}(p) = 0$. Now, $\mathcal{L}(p) = \sum \beta_i \mathcal{L}(t_i) \leq_{\mathbb{P}} \sum \beta_i R_{\mathcal{T}}(t_i) = R_{\mathcal{T}}(p) = 0$. For any axiom M_v let $M_v = t + w$, where t is the leading term. Since $\Gamma(v) \subseteq Vertex(t)$, then $v \in Sup(t)$. Hence $\mathcal{L}(v) = \mathcal{L}(t) + \mathcal{L}(w) \leq_{\mathbb{P}} R_{M_v}(t) + \mathcal{L}(w) = -w + \mathcal{L}(w) \leq_{\mathbb{P}} -w + w = 0$.

Let us prove that $\mathcal{L}(xt) = \mathcal{L}(x\mathcal{L}(t))$ for any term t of degree strictly less than $\frac{cr}{4}$. Notice that by monotonicity of Sup function, $Sup(xt) \supseteq Sup(t)$. Moreover since $deg(xt) \leq \frac{cr}{4}$, then by Lemma 3.6 we get $|Sup(xt)| \leq r/2$. Therefore we have the following chain of equalities by applying respectively: in (3.4) the definition; in (3.5) by Fact 2.2; in (3.6) the monotonicity of Sup and Lemma 3.9; in (3.7) again the definition.

$$\begin{aligned} \mathcal{L}(xt) &= R_{\mathcal{T},M_{Sup(xt)}}(xt) && (3.4)\\ &= R_{\mathcal{T},M_{Sup(xt)}}(xR_{\mathcal{T},M_{Sup(xt)}}(t)) && (3.5)\\ &= R_{\mathcal{T},M_{Sup(xt)}}(xR_{\mathcal{T},M_{Sup(t)}}(t)) && (3.6)\\ &= R_{\mathcal{T},M_{Sup(xt)}}(x\mathcal{L}(t)) && (3.7) \end{aligned}$$

Let us write $x\mathcal{L}(t)$ as a polynomial $\sum \alpha_i r_i$. The following inclusions hold respectively: in (3.8) because r_i is a monomial in the polynomial expansion of $x\mathcal{L}(t)$; in (3.9) by Lemma 3.10; in (3.10) by monotonicity of Sup.

$$\begin{aligned} Vertex(r_i) &\subseteq Vertex(x) \cup Vertex(\mathcal{L}(t)) && (3.8)\\ &\subseteq Vertex(x) \cup Vertex(t) \cup Sup(t) && (3.9)\\ &\subseteq Vertex(xt) \cup Sup(xt) && (3.10) \end{aligned}$$

From the definition of Sup and the previous inclusions it follows that $Sup(r_i) \subseteq Sup(xt)$.

Finally the property (2) of the operator is obtained from the following chain of equalities respectively motivated: in (3.11) by definition; in (3.12) by Lemma 3.9 applied to $Sup(r_i)$ and $Sup(xt)$; in (3.13) by linearity; in (3.14) by the form of $x\mathcal{L}(t)$; finally in (3.15) by equalities (3.4)-(3.7).

$$\begin{aligned} \mathcal{L}(x\mathcal{L}(t)) &= \sum \alpha_i R_{\mathcal{T},M_{Sup(r_i)}}(r_i) && (3.11)\\ &= \sum \alpha_i R_{\mathcal{T},M_{Sup(xt)}}(r_i) && (3.12)\\ &= R_{\mathcal{T},M_{Sup(xt)}}(\sum \alpha_i r_i) && (3.13)\\ &= R_{\mathcal{T},M_{Sup(xt)}}(x\mathcal{L}(t)) && (3.14)\\ &= \mathcal{L}(xt) && (3.15) \end{aligned}$$

Finally for the property (3) observe that the support of a constant is the empty set, so $\mathcal{L}(1) = R_{\mathcal{T}}(1) = 1$ since $\mathcal{T}$ is satisfiable. $\square$

To conclude the proof of Theorem 3.2 is sufficient to exhibit a uniform family $\mathcal{G} = \{G_n\}_{n\in\mathbb{N}}$ of graph with $\Theta(n)$ vertices, constant degree such that any member is a $(\Theta(n), \Omega(1))$ vertex expander. Such families are known in literature [36].

Relation with size-degree tradeoffs

In Chapter 2 we discuss a known relation among size and degree for refutations in Polynomial Calculus and Polynomial Calculus with Resolution. We recap the tradeoff

$$S \geq 2^{\Theta\frac{(D-d)^2}{v}} \qquad\qquad D \leq O(\sqrt(v \log S)) + d$$

where S and D is respectively the smallest size and degree of a refutation of the theorem, and d and v are respectively the degree and the variables of its polynomial formulation. $\textsc{Gop}(G)$ is tight in term of the exponent because it has $v = \Theta(n^2)$, $D = \Theta(n)$, $d = O(1)$ and $S = n^{O(1)}$. No better strategy exists to reduce the degree of a proof than the one described in Chapter 2.

This trade-off is trivially optimal for principles with exponential size proofs: Razborov in [47] showed that any $\textsc{Pc}$ refutation for PHP^m_n requires degree n. PHP^m_n has also a trivial proof of size $2^{O(n)}$. Bonet and Galesi in [18, 19] asked to prove the trade-off optimality for formulas having polynomial size Resolution refutations. They had partial results in this direction: they proved that a modification of the pigeon hole principle has efficient refutations in Resolution but requires degree $O(\log n)$ [20]. Our result on $\textsc{Gop}(G)$ exponentially improves the results of [20] showing a linear degree lower bound for a formula efficiently refutable in Resolution, then closing the problem left open in [18, 19].

Chapter 4

Non Automatizability of Polynomial Calculus

Automated theorem proving is one of the most important field in Computer Science, if not the application computers were invented for. It has been investigated both from a theoretical and an applied point of view. Not only it is widely conjectured that for any proof system there are tautologies requiring proof of exponentially long proofs. But also it is believed that there are tautologies whose short proofs elude any efficient algorithm. This is related with the concept of automatization which has been introduced in Chapter 1.

We prove that Polynomial Calculus and Polynomial Calculus with Resolution are not automatizable unless **W[P]**-hard problems are tractable in a specific sense. This extends to Polynomial Calculus the analogous result obtained for Resolution by Alekhnovich and Razborov [8].

In Chapter 2 we discussed about the inefficiency of the Buchberger algorithm for tautologies which require high degree proofs. That was a very peculiar algorithm. Now we consider any kind of proof searching algorithms for a system PC, PCR and HN.

The result comes from the reduction of Alekhnovich and Razborov [8]. We extract the part of their proof which actually depends on the RES proof system and we adapt such part to our proof systems.

The key step is a degree lower bound for a formula encoding the decision version of the problem known as *minimum monotone circuit satisfying assignment* (MMCSA). We refer the reader to Section B.3 for more details about this problem and its relation with Parameterized Computational Complexity. We prove the needed degree lower bound with the technique seen in Chapter 3.

The chapter is organized as follows: in Section 4.1 we give a sketch of the non automatizability proof; in Section 4.2 we give the polynomial encoding of MMCSA; Section 4.3 contains the proof of the degree lower bound; finally in Section 4.4 we combine together the degree lower bound with previous results of [8] to get the non automatizability of HN, PC and PCR.

4.1 Proof Strategy

The optimization problem of *Minimum Monotone Circuit Satisfying Assignment* (MMCSA) is defined as follows

Instance A monotone circuit C over $\wedge, \vee$ in n variables with 0-1 variables.

Solution An input a such that $C(a) = 1$

Objective function Minimize $w(a)$, the Hamming weight of a.

We denote as $w(C)$ the minimum for the instance C. In [8] Alekhnovich and Razborov use the automatization of Resolution as a primitive to efficiently solve MMCSA. The idea can be summarized in three independent steps.

1. Given a monotone circuit C, build an unsatisfiable CNF $F(C, w, r)$ and prove that the size of the shorter proof is strongly related to the size of minimum satisfying assignment of the circuit.

2. Assuming automatizability of the proof systems, use the automatization algorithm to find a proof of approximately small size. This gives an approximation of the minimum assignment size.

3. Apply a (randomized) gap amplification procedure to improve the approximation factor up to an error smaller than one (i.e. obtain the exact value).

Only the first step depends on the proof system. Using a slight modification of the formula built in [8] we will prove the first step for algebraic proof systems. In the end we are going to prove the following theorems.

Theorem 4.1. *If any of* HN, Pc *or* Pcr *is automatizable, then for a fixed $\epsilon > 0$ there exists an algorithm Φ working on monotone circuits C which runs in time $\exp\left(w(C)^{O(1)}\right)|C|^{O(1)}$ and approximates the value of $w(C)$ to within a factor $(1 + \epsilon)$.*

Theorem 4.2. *If* HN, Pc *or* Pcr *are automatizable then* **W[P]**=**coFPR**.

W[P] and **coFPR** are well-known complexity classes from parameterized computational complexity theory (see Section B.3 or the book [29]). The decision version of MMCSA is complete for **W[P]**, so it seems unlikely that there could by a fixed parameter algorithm for it.

To get Theorem 4.1 we prove the following reduction as in [8]: from a circuit C we define the formula $F(C, w, r)$ such that next two lemmas hold.

Lemma 4.3. *Let C be a monotone circuit, and w an integer parameter. Any* Pcr *refutation of $F(C, w, r)$ requires degree at least $r \min\{w(C) - 1, w\}$.*

Lemma 4.4. *Let C be a monotone circuit, and w an integer parameter. Assume $r = \Theta(w)$:*

1. *Any* Pcr *refutation of $F(C, w, r)$ has size at least*

$$2^{\Omega(w \min\{w(C),w\})}$$

2. *If $w(C) \leq w$ then there is a* HN *proof of $F(C, w, r)$ of size*

$$|C| \cdot 2^{O(w \cdot w(C))}$$

4.2 The tautology $F(C, w, r)$

Let C be a monotone circuit on n inputs of size polynomial in n. Let w be a parameter whose intended meaning is to guess a value for the value of $w(C)$. We also fix r to be a parameter which will be used to amplify the degree hardness of the principle. Such r is intended to be $\Theta(w)$.

A combinatorial object called *Paley matrix* is used in the construction. Such matrix has the property that any projection on a small set of columns consists of all possible binary strings with an almost fair frequency. Here follows an example of a 7×7 matrix where any projection on 2 rows or columns contains $\{00, 01, 10, 11\}$.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Let $q \in [2^{4w} + 1, 2^{4w+1}]$ be an arbitrary prime[1], and consider the matrix $A \in \{0,1\}^{q \times q}$ where cell (i, j) contains 1 iff $i - j$ is a quadratic residue modulo q, and 0 otherwise[2]. It is well known [39, 9] that any projection on w rows or columns contains the whole set $\{0,1\}^w$.

From C, w, r and A, we are going to define a set of polynomials $F(C, w, r)$. This set encodes the property that there exists a set of n columns of the matrix A that contains no satisfying assignment for the circuit C. This claim is unsatisfiable if $w(C) < w$ because whatever the selection of n columns is, the $w(C)$ columns corresponding to the 1 coordinates of the smallest assignment will contain a full 1 string which satisfies the circuit C.

The n columns are selected by n partial functions called *generators*. These functions will be defined later.

For technical reasons, widely explained in [8], we add complexity to the principle: instead of feeding inputs to a single circuit, we feed them to several copies of the same circuit, and we select one of those copies by a **partial** function called *activator*. In the principle we will express the fact that only the gates of the active circuit has to perform the computation correctly.

The principle claims that exists a choice for the 'generators and activators' such that: (a) values for such inputs are defined and (b) no circuit output 1.

It is clear that even in this version of the principle $w(C) < w$ implies that for any choice the w generators produce a row with w ones in the appropriate coordinates. This row is fed to a set of r copies of the circuit C. One of them is active and has to propagate the computation through the gates. That circuit copy outputs 1, which is in contradiction with the claim no active circuit output 1.

Let s be a parameter we will specify later and is intended to be $\Theta(w)$.

Circuit family: let $q = 2^{\Theta(w)}$ the size of the matrix A. We consider an array of $q \times r$ copies of circuit C, indexed as C_{ik}.

Activators: for all $i \in [q]$ we consider a possibly partial function $A_i : \{0,1\}^s \mapsto [r]$ which selects one of the circuit among $C_{i1} \cdots C_{ir}$. Only the selected circuit propagates correctly the values through the gates. The other circuits are relieved from the computation.

Generators: for $j \in [n]$ we consider $G_j : \{0,1\}^s \to \{0,1\}^q$ which generates a column vector of the matrix A. Thus $\{G_1 \cdots G_n\}$ produce the $q \times n$ matrix A. For all $i \in [q]$ the i-th row of such matrix is fed to $C_{i1} \cdots C_{ir}$ as input.

Definition 4.5. *A boolean function f is r-**surjective** if for any partial assignment ρ such that $|\rho| \leq r$ the function $f\lceil_\rho$ is surjective.*

Fact 4.6. *For any β and any surjective function $g : \{0,1\}^l \to S$ there exists a βl-surjective function $f : \{0,1\}^{O(\beta l)} \to S$.*

[1]Such a prime exists because of Bertrand's postulate.

[2]Note that in literature the original Paley Matrix is defined in a different fashion with 0 values on diagonal, 1 for quadratic residues and -1 for non residues. But this is not an issue here

Proof. Consider a surjective decoding function $D : \{0,1\}^{\alpha l} \longrightarrow \{0,1\}^{l}$ of a code with distance $\geq 2\beta l$. Such decoder exists [54] with α proportional to 2β. Fix $f := g \circ D$, it satisfies the requirements. To see this consider the restricted βl bits as "error" bits. Any message can be decoded from the proper codeword with those βl bits changed. $\square$

Generator and Activators with r-surjectivity and parameter s. We need both activators and generators to be r-surjective functions on domain $\{0,1\}^{s}$. See [8] for an extensive discussion about this requirement.

The output of a generator function G_j is completely specified by the choice of a column in the Paley matrix, the output of the activator A_i is specified by the choice of a number in $[r]$. Thus $\Theta(\max\{w, \log r\})$ input bits are enough to define both surjective generators and selectors. Then by Fact 4.6 we know $s = \Theta(w + r)$ is more than enough to have the desired r-surjective activators and generators.

Because of the properties of the Paley matrix, of the generators and of the activators the following fact holds.

Fact 4.7. *For any monotone circuit C, and any $w > w(C)$, at least a circuit among $\{C_{ik}\}_{ik}$ outputs 1 whatever the assignments for activators and generators are.*

4.2.1 Polynomial encoding of $F(C, w, r)$

We now describe the actual set of polynomials which encode the negation of Fact 4.7. We will use the following notation.

Notation reference

C	the monotone circuit.
n	input variables in the circuit C.
v	index refers to a the gates of C. It goes from 1 to $\lvert C\rvert$. We assume that first $1..n$ gates correspond to input gates. Gate indexed by $\lvert C\rvert$ corresponds to output gates.
$w(C)$	is the minimum hamming weight of an assignment satisfying C.
$q = 2^{\Theta(w)}$	a prime number, size of the Paley matrix.
r	is a parameter.
i	index used to denote a row in the Paley matrix, a selector function and an input fed to the circuits. It goes from 1 to q.
j	index used to denote an input variable of C and one of the generator functions. It goes from 1 to n.
k	index used to denote one of the possible outcomes of the selector functions. It goes from 1 to r.
C_{ik}	k-th copy of C, fed with the i-th row generated by generator functions.
A_i	i-th activator function which selects among circuits $C_{i1}, \ldots, C_{ir}$.
G_j	j-th generator function which choose the j-th columns to feed as j-th input of the circuits.
s	length of the binary input of activators and generators, it is $\Theta(w + r)$.

$x_j(\alpha), y_i(\beta)$ For a vector $\alpha, \beta \in \{0,1\}^s$ we denote as $x_j(\alpha)$ and $y_i(\beta)$ the characteristic polynomials of α and β respectively on variables $x_{j1}, \ldots, x_{js}$ and $y_{i1}, \ldots, y_{is}$. For example $x_j(001\ldots)$ is $\bar{x}_{j1}\bar{x}_{j2}x_{j3}\cdots$.

The polynomial encoding of the principle is expressed in the following sets of variables.
Variables For all i, j, k, and gate v as above:

- $x_{j1} \ldots x_{js}$ are the variables representing the input of generator G_j.
- $y_{i1} \ldots y_{is}$ are the variables representing the input of activator A_i.
- z^v_{ik} represents the value of gate v in circuit C_{ik}.

We now give the encoding. For any index $i \in [q]$ we group in a set F_i the polynomials relevant to a row i. Indexes i, j, k, v and α, β apply properly as described above. The polynomials in a set F_i are shown below:

$$
\begin{aligned}
&x_j(\alpha) \cdot y_i(\beta) \cdot \bar{z}^j_{i,k} && \text{when the } i^{th} \text{ bit of } G_j(\alpha) \text{ is 1 and } A_i(\beta) = k && (4.1)\\
&y_i(\beta) && \text{when } \beta \notin dom(A_i) && (4.2)\\
&y_i(\beta) \cdot z^A_{i,k} \cdot z^B_{i,k} \cdot \bar{z}^v_{i,k} && \text{gate } v \leftarrow A \wedge B \text{ and } A_i(\beta) = k && (4.3)\\
&\left.\begin{array}{l} y_i(\beta) \cdot z^A_{i,k} \cdot \bar{z}^v_{i,k} \\ y_i(\beta) \cdot z^B_{i,k} \cdot \bar{z}^v_{i,k} \end{array}\right\} && \text{gate } Y \leftarrow A \vee B, \text{ and } A_i(\beta) = k && (4.4)\\
&y_i(\beta) z^{|C|}_{i,k} && \text{when } A_i(\beta) = k && (4.5)
\end{aligned}
$$

Consider a common root for these polynomials: (4.1) says that if the i-th bit of the column generated by G_j is 1 and the active circuits for i-th row is k, then the j-th input of C_{ik} must be on. (4.2) forces the y_i variables to encode a value in the domain of the activator A_i. This is necessary because activators are partial functions. (4.3) and (4.4) force the active circuit to compute the gates correctly. The equation (4.5) claims that the output of the active circuit is zero.

The principle $F(C, w, r)$ consists in the conjunction of F_i for $i \in [q]$ plus the canonical boolean axioms of PCR.

Notice that this principle is slightly different from the one in [8], where additional variables are used to encode circuits activation. We choose not to use them because they would cause troubles in some technical steps of the following proofs.

Definition 4.8. *We call $F(C, w, r)$ the principle described above as $\bigcup_i F_i$, where C is a monotone circuit, $q = 2^{\Theta(w)}$ is a prime, and r is the surjectivity parameter of generators and activators.*

Fact 4.9. *The polynomials of principle $F(C, w, r)$ can be produced in $|C|2^{O(w+r)}$ time and space. If $w(C) \leq w$ then $F(C, w, r)$ is unsatisfiable.*

Proof. Remember $s = O(w+r)$. For any α, β two strings of s bits, any $i \in [q]$ and any $j \in [n]$ either there is exactly one k such that polynomial $x_j(\alpha)y_i(\beta)\bar{z}^j_{i,k} \in F_i$, or there is a polynomial $y_i(\beta) \in F_i$. In total we have $n2^{O(s)} + |C|2^{O(s)})$ polynomials. Notice that all of them are monomials. Even if the encoding does not use negated variables, the expanded polynomials would be exponentially long in the degree of the equation. Biggest degree appearing is $2s$ thus the size of the formula $F(C, w, r)$ is again $n2^{O(s)} + |C|2^{O(s)}$. The obvious output strategy satisfies the resource bound. Unsatisfiability comes as a restatement of Fact 4.7. □

4.3 Degree Lower bounds for $F(C, w, r)$

In this section we prove the formula $F(C, w, r)$ requires a high degree to be refuted. We need a tailored degree measure for this purpose.

Definition 4.10. *For a monomial t consider the three sets*

$$\begin{aligned} X_t &:= \{(j,l) : x_{jl} \in t\} \\ Y_t &:= \{(i,l) : y_{il} \in t\} \\ Z_t &:= \{(i,k) : \text{there is a } v \text{ for which } z^v_{ik} \in t\} \end{aligned}$$

We define the ***index-degree*** *of t as*

$$\mathrm{ideg}(t) = |X_t| + |Y_t| + |Z_t|$$

The index-degree of a polynomial is the biggest index-degree among its monomials.

Lemma 4.11. *Let C be a monotone circuit, and w an integer parameter. Any* Pcr *refutation of $F(C, w, r)$ requires degree at least $r \cdot \min\{w(C) - 1, w\}$.*

From now on we fix $m := min\{w(C) - 1, w\}$. The index-degree lower bound relies on the construction of an operator K over multivariate polynomials such that

1. K is a linear operator.
2. $K(p) = 0$ for any p in $F(C, w, r)$.
3. If $\mathrm{ideg}(t) < rm$ then $K(xt) = K(xK(t))$ holds for any variable x.
4. $K(1) \neq 0$

Proof. (of Lemma 4.11) Assume a proof of index-degree less than rm exists: each line of such proof is either an equation in $F(C, w, r)$, or a sum of previous lines, or the product of a previous line with a variable where index-degree stays below rm. Then property (1), (2), (3) imply that K maps to 0 every line in the proof. This contradicts property (4) which claim K can not map last line to 0. □

We now exhibit such operator K. Assume $I \subseteq [q]$ and consider the set of polynomials $\mathcal{S}$ the union of $\bigcup_{i \in I} F_i$ and of all Pcr axioms. In the rest of the chapter we abuse the notation regarding $I \in [q]$ in this way:

$$\begin{array}{rcl} R_I(p) & := & R_{\mathcal{S}}(p) \\ I \vdash p & \text{iff} & \mathcal{S} \vdash p \\ I\restriction_\rho & \text{is the set} & \{p\restriction_\rho \text{ such that } p \in \mathcal{S}\} \\ A \subseteq I & \text{for any polynomial set } A \text{ if} & A \subseteq \mathcal{S} \\ I \subseteq B & \text{for any polynomial set } B \text{ if} & \mathcal{S} \subseteq B \end{array}$$

where R is the usual residue in the polynomial ring (see Definition 2.1) with respect to a set of polynomials, and $\vdash$ is the usual Pc deduction from a set of polynomials. The meaning of this notation abuse is to identify a set $I \in [q]$ with its induced polynomial set.

Definition 4.12. Function I and operator K:

Fix a monomial t: we can write $t = t_1 t_2 \cdots t_q t'$ where each t_i contains only variables indexed by $i \in [q]$ and t' contains only generator variables of type x_{jl}. We define

$I(t)$ as the set of $i \in [q]$ such that $\mathrm{ideg}(t_i) \geq r$.

$K(t)$ to be equal to $R_{I(t)}(t)$.

On a polynomials $p = \sum_i c_i t_i$, $K(p)$ is defined to be equal to $\sum_i c_i K(t_i)$.

We now check that K satisfies properties (1)-(4). (1) It comes from the definition. (2) If a premise p is an axiom of the system then any of its term is reduced with respect to an ideal which contains p itself. Any premise p in F_i is a monomial t which contains more than r variables indexed by i. Thus such p is in the ideal induced by $I(t)$. This implies $K(p) = 0$. (4) is true because $I(1)$ induces a set of polynomials with a common solution.

To prove (3) we need the two following properties of ideals:

Lemma 4.13. *For any polynomial p and ideal J generated by $q_1, q_2, \ldots, q_m$, all variables appearing in $R_J(p)$ also occur in $p, q_1, q_2, \ldots, q_m$.*

Proof. Let x be a variable occurring in $R_J(p)$ and neither in p nor in any q_i. By definition $p - R_J(p) = \sum_i h_i q_i$ for some polynomials h_i thus by setting x to 0 we obtain $p - R_J(p) \restriction_{x=0} = \sum_i h_i \restriction_{x=0} q_i$ where $R_J(p) \restriction_{x=0}$ is strictly smaller than $R_J(p)$. This contradicts that $R_J(p)$ is the minimum according to $<_{\mathbb{P}}$. □

Corollary 4.14. *Let m_1, m_2 two monomials, if t is a monomial in $m_1 \cdot R_{I(m_2)}(m_2)$ then $I(t) \subseteq I(m_1 m_2)$.*

Proof. If $i' \in I(t)/I(m_1 m_2)$ then variables indexed by i' in t are more than r. No such variable is contained in F_i for $i' \neq i$. Thus by Lemma 4.13 any term in $R_{I(m_2)}(m_2)$ does not contain more of such variables than m_2 itself contains. Such i'-indexed variables are then contained in the union of the variable of m_1 and m_2. Thus $i' \in I(m_1 m_2)$ but this contradicts the assumption. □

Next lemma is the heart of the argument: it shows how a small index-degree derivation has local behavior. The set of premises needed in the derivation is a subset of the one given by the operator I.

Lemma 4.15. *Let t be a monomial of index-degree less than rm and $I(t) \subseteq I \subseteq [q]$ with $|I| \leq m$. Then $R_{I(t)}(t) = R_I(t)$.*

Proof. We will show an assignment ρ such that $t \restriction_\rho = t$ and $I \restriction_\rho \subseteq I(t)$. This is sufficient because if

$$I \restriction_\rho \vdash t \restriction_\rho - R_I(t) \restriction_\rho$$

then by properties of ρ we get $I(t) \vdash t - R_I(t) \restriction_\rho$. This means $R_I(t) \restriction_\rho$ is bigger than $R_{I(t)}(t)$ in the order among polynomials. It is also smaller than $R_I(t)$ because a partial assignment can't increase the order. Notice that we also have $R_I(t)$ smaller then $R_{I(t)}(t)$ because $I(t)$ is a subset of I and residue is monotone decreasing with respect to the subscript set. Thus $R_I(t) = R_{I(t)}(t)$.

To construct such ρ we now consider J the set in indexes $j \in [n]$ such that t contains less r variables among $x_{j1}, \ldots, x_{js}$. Thus $|[n] - J| \leq m$.

Notice that because of r-surjectivity of generators we have that for any $j \in J$ and any vector $v = v_1 \ldots v_q$ in the image of G_j there is a boolean partial assignment α_j on "x_j" variables such that no variable in t is assigned and $G_j(\alpha) = v$. We choose a v such that for any i in I we have $v_i = 0$. Such choice is possible because $|I| \leq m$ and v is a column in a Paley matrix of appropriate size. We add partial assignments α_j for $j \in J$ to ρ. Such partial assignments do not restrict t, and set to 0 all polynomials $x_j(\alpha) y_i(\beta) \bar{z}^j_{ik}$ for any $j \in J$, $i \in I$, $k \in [q]$. Other polynomials have not been touched so far.

We now consider a row i_0 in $I/I(t)$ and t_0 the monomial containing all variables in t indexed by i_0. We extend ρ to restrict to 0 all remaining polynomials in F_{i_0} without restricting t. There is at least one circuit copy C_{i_0k} such that no variables in t correspond to a gate of such circuit, otherwise it would be $\mathrm{ideg}(t_0) \geq r$ and i_0 would be in $I(t)$. For the same reason we also know in t there are less than r variables among $y_{i_01} \dots y_{i_0s}$. Both observations together imply there is a partial assignment to the y_{i_0} variables not contained in t such that $y_{i_0}(\beta) = 0$ for all β with $A_{i_0}(\beta) \neq k$. Then by now all polynomials in F_{i_0} are satisfied with the exception of the ones corresponding to circuit C_{i_0k}. We set $z^j_{i_0k}$ to 0 when $j \in J$ and 1 otherwise. Then we propagate values among the circuit accordingly. We remark that being $|J| < m \leq w(C)$ we have 0 at the output gate. This satisfy all clauses in F_{i_0k} without touching t. We continue to extend ρ in this way for all for all $i \in I/I(t)$. The resulting assignment satisfies the requested properties. Thus the lemma is proved. □

Lemma 4.16. *If the index-degree of a monomial t is less than rm then $K(xt) = K(xK(t))$*

Proof. Consider a monomial t of index-degree less than rm. We will prove that both $K(xt)$ and $K(xK(t))$ are equal to $R_{I(xt)}(xK(t))$. Consider the following chain of equations.

$$\begin{aligned} K(xt) = && R_{I(xt)}(xt) && (4.6)\\ = && R_{I(xt)}(xR_{I(xt)}(t)) && (4.7)\\ = && R_{I(xt)}(xR_{I(t)}(t)) && (4.8)\\ = && R_{I(xt)}(xK(t)) && (4.9)\end{aligned}$$

The equation (4.6) is the definition; (4.7) because R_I operator is an homomorphism on the ring of multivariate polynomials; (4.8) holds because $|I(xt)| \leq |I(t)| + 1 \leq m$ and Lemma 4.15 applies; (4.9) holds because of the definition of K. Let us denote $xK(t)$ as $\sum_i \alpha_i t_i$ in the next chain of equations.

$$\begin{aligned} K(xK(t)) = && K(\sum_i \alpha_i t_i) && (4.10)\\ = && \sum_i \alpha_i K(t_i) && (4.11)\\ = && \sum_i \alpha_i R_{I(t_i)}(t_i) && (4.12)\\ = && \sum_i \alpha_i R_{I(xt)}(t_i) && (4.13)\\ = && R_{I(xt)}(\sum_i \alpha_i t_i) && (4.14)\\ = && R_{I(xt)}(xK(t)) && (4.15)\end{aligned}$$

The first lines holds because the notation just introduced; (4.11) by linearity of K; (4.12) by definition of K; (4.13) holds because any t_i is a monomial in $xR_{I(t)}(t)$. We now use Corollary 4.14 to claim $I(t_i)$ is a subset of $I(xt)$, which has size less than m. Lemma 4.13 finally implies the equation. By using linearity we get (4.14) and by reverting the change of notation we conclude the proof with equation (4.15). □

4.4 Non automatizability of HN, Pc and Pcr

In this section we prove a result similar to Lemma 3.1 of [8], for proof systems HN, Pc and Pcr. Results obtained in Section 3 of [8] depend on Resolution system, while the self-improvement technique developed in Section 4 of [8] refers to MMCSA amplification and is independent from the proof system adopted.

Lemma 4.17. *Let C be a monotone circuit, and w an integer parameter. Assume that $r = \Theta(w)$:*

1. *Any* Pcr *refutation of $F(C, w, r)$ has size at least*
$$2^{\Omega(r \min\{w(C),w\})}$$

2. *If $w(C) \leq w$ then there is a* HN *proof of $F(C, w, r)$ of size*
$$|C| \cdot 2^{O(r \cdot w(C))}$$

Proof. **(1) Lower bound.** The strategy here follows [8]: we deduce a degree lower bound on the Pcr refutation of $F(C, w, r)$ and then we use a random restriction/probabilistic method argument to deduce the size lower bound.

The restriction: for each generator and activator we restrict uniformly independently at random a set of $r/2$ of the s variables in each input set. For each $i \in [q]$ we also choose independently $r/2$ circuit copies of the r available and we restrict randomly all the gates of such copies. The restricted polynomial set is essentially (up to index reordering) subsumed by $F(C, w, r/2)$.

Fix $d := \frac{r}{4} \cdot \min\{w(C)-1, w\}$. We show that any monomial with index-degree bigger that d is set to zero with probability at least $1-2^{-\Omega(d)}$. Fix a polynomial t of index degree at least d. We factor $t = t_1 \cdots t_d t'$ where t_i is either a generator variable, an activator variable or a non empty product of gate variables corresponding to a particular circuit C_{ik}. We want to estimate the probability that t_i is set to 0 by the random restriction, assuming $t_1 \ldots t_{i-1}$ haven't been. Consider the case t_i is a generator or an activator variable: s is the number of such variables for each generator and activator. With at least $r/2s$ probability t_i is chosen among the restricted variables. Then with probability at least $r/4s$ the monomial is set to zero. We have $r = \Theta(w)$ by assumption and $s = \Theta(r+w)$ by construction, then $r/4s$ is a constant. In the case t_i is a product of variables of C_{ik} for some i and k then such circuit is chosen to be restricted with probability at least $1/2$ because no previous one has been, and the product is restricted to zero with at least probability $1/4$. The probability of the monomial not to be set to zero is then at most c^d for some constant $c < 1$.

For a partial assignment ρ distributed as described $\Pi\restriction_\rho$ is a refutation of $F(C, w, r)\restriction_\rho$. Assuming there exists Π of size smaller than c^d then by union bound there is a restriction ρ such that $\Pi\restriction_\rho$ is a refutation of degree less than d for $F(C, w, r)\restriction_\rho$. By a simple reordering and weakening we obtain a refutation for $F(C, w, r/2)$. This is in contradiction with the index-degree lower bound proved in Section 4.3.
(2) Upper bound. In this hypothesis the principle is unsatisfiable because of Fact 4.9. In this case a tree-like refutation of size $|C|2^{O(s \cdot w(C))}$ for $F(C, w, r)$ exists as it is shown in [8]. Such proof can be simulated in Pcand PcrEasily. For Pcthe absence of dual variables leads to manipulate big representations of polynomials, but the asymptotic complexity of the proof stays the same. For completeness we also show a proof in HN.

We now assume wlog the first $1 \ldots w(C)$ inputs correspond to the minimum satisfying assignment.

We have to prove there are multiples of premises which sum up to 1. Notice that by definition of characteristic functions we have $1 = \sum_{\alpha \in \{0,1\}^s} X_j(\alpha)$ for any $j \in [n]$ and also $1 = \sum_{\beta \in \{0,1\}^s} Y_i(\beta)$

for any $i \in [q]$. Then we get

$$1 = \sum_{\alpha_1 \ldots \alpha_{w(C)} \beta} X_1(\alpha_1) \cdots X_{w(C)}(\alpha_{w(C)}) Y_i(\beta)$$

for any i, in particular we fix $i := i(\alpha_1, \ldots, \alpha_{w(C)})$ to be a row containing a satisfying assignment generated by $\alpha_1 \ldots \alpha_{w(C)}$. This immediately implies there is a value k for which C_{ik} outputs 1. Fix $p_0 := X_1(\alpha_1) \cdots X_{w(C)}(\alpha_{w(C)}) Y_i(\beta)$ be one of the polynomials in the sum, and let be k the corresponding activated circuit.

We now show p_0 can be written as sum of premises: consider the propagation of the satisfying assignment through C_{ik} (from now on we drop the ik indexes for sake of notation). There is a minimal sequence of gates $z^1 \ldots z^m$ in the circuit such that z^m is the output gate, $z^1 \ldots z^{w(C)}$ are the input gates activated by generators, for any AND gate both its inputs are predecessors in the sequence, for any OR gate at least one of its input is also a predecessor in the sequence. We denote $p_l := p_0 z^1 \cdots z^l$. We prove by backward induction on l that p_l is provable in Hilbert Nullstellensatz.

Base case: p_m is a multiple of $Y_i(\beta) z^m_{ik}$ which is a premise.

Induction step: assuming p_l is provable. By minimality the gate z^l is activated by some predecessor(s) in the sequence. Then $p_{l-1} = p_{l-1}(1 - z^l - \bar{z}^l) + p_{l-1}\bar{z}^l + p_{l-1} z^l$. The first part comes from boolean axioms, the second part is a multiple of $Y_i(\beta) z^A_{ik} z^l_{ik}$ ($Y_i(\beta) z^A_{ik} z^B_{ik} z^l_{ik}$) if the gate is an OR (respectively an AND), the third part comes from inductive hypothesis.

Then p_0 can be proved in $|C|^{O(1)}$. The number of such polynomials to prove are $2^{s \cdot w(C) + s}$.

To prove that the sum of characteristic functions gives 1 it is sufficient an extensive use of boolean axioms of dual variable. This leads to a proof of size $|C|^{O(1)} \cdot 2^{s \cdot w(C)+s} + 2^{O(s \cdot w(C)+s)}$. Because $r = \Theta(w)$ we get $s = \Theta(r + w) = \Theta(w)$ and the final claim.

□

Lemma 3.1 of [8] can be now be rephrased for HN and PCR, as follows

Lemma 4.18. *There exists a polynomial time computable function τ which maps any pair $\langle C, 1^m \rangle$, where C is a monotone circuit and m is an integer into an unsatisfiable CNF $\tau(C, m)$ such that:*

- *there is a* HN *proof of $\tau(C, m)$ of size $|C| m^{O(\min\{w(C), \log m\})}$*
- *Any* PCR *refutations of $\tau(C, m)$ has size at least $m^{\Omega(\min\{w(C), \log m\})}$*

Proof. Set $w = \log m/4$ and $r = \Theta(w)$ according to Lemma 4.17. Define

$$\tau(C, m) := F(C, w, r) \wedge \tau_m$$

where τ_m is the pigeon hole principle PHP^{l+1}_l where $l = \log^2 m$, plus $p_{i,j} + \bar{p}_{i,j} - 1$ for $(i, j) \in [l + 1] \times [l]$. The p_{ij} variables of pigeon hole principle are disjoint from $F(C, w, r)$ ones.

The size lower bound $m^{\Omega(\min\{w(C), \log m\})}$ holds for both τ_m (by [47, 37]) and $F(C, w, r)$ (by Lemma 4.17). Then holds for $\tau(C, m)$ because Feasible Interpolation and the Weak Feasible Disjunction properties hold for Polynomial Calculus (see [45, 46]). More concretely from a short proof of $\tau(C, m)$ we could extract a short proof of τ_m or $F(C, w, r)$.

The upper bound holds because if $w(C) < \frac{\log m}{4}$ then we can use the upper bound in Lemma 4.17, otherwise we can refute τ_m in size $2^{\log^2 m}$ because PHP^{n+1}_n has $2^{O(n)}$ treelike Resolution proof and HN polynomially simulates treelike Resolution. □

We conclude the chapter by sketching the proof of Theorem 4.1 and 4.2.

Proof. (Theorem 4.1 and 4.2)

Theorem 4.1 and 4.2 are essentially a rephrasing of Theorem 2.5 and Theorem 2.7 in [8], where proof systems HN, Pc and Pcr substitute Resolution. In their proof the only part depending on the proof system is Lemma 3.1 of [8]. Such lemma for HN, Pc, Pcr is exactly our Lemma 4.17. Then the proof of both Theorem follows in the same way as in [8]. □

Chapter 5

PCR_k proof system

In this chapter we extend the PCR system. We define the system PCR_k by combining PC and RES_k. In the whole chapter we assume $k = O(1)$ (our results hold for any k less than $\sqrt{\frac{\log n}{\log \log n}}$, but we fix it to a constant to simplify the proofs). Monomials in PCR succinctly represent clauses by using dual variables to express negation. Notice that the product of n factors like $(1 - x_i)$ is a polynomial with 2^n terms: a clause with many negations requires a large polynomial representation. In PCR the problem is solved by using $\bar{x}_i$ as a placeholder for $(1 - x_i)$. In this way any clause has a PCR representation of linear size. Placeholders in a monomial can be efficiently expanded by using the axiom $1 - x_i - \bar{x}_i$.

We are going to define a similar extension of PC in which k-DNF can be efficiently represented. Instead of adding placeholders, we allow lines in the proof to be more complex algebraic expressions. This proof system is called PCR_k.

In this chapter we show that the degree complexity required to prove a theorem in PCR_k is essentially the same required in PC, i.e. PC simulates PCR_k in the same degree. Degree is not affected by the use of stronger formulas, but size could.

We analyze the power of PCR_k: we prove that PCR_k efficiently simulates RES_k, and is exponentially stronger than RES_k on certain principles. To do this we define a variant of the matching principle which has short proof in PCR_k and requires exponential size in RES_k. It turns out that such variant of matching principle is efficiently provable even in PCR. Even if PCR_k is stronger than RES_k it can not prove the regular (weak) pigeonhole principle in polynomial size. This follows immediately from the probabilistic argument used in [50] for RES_k. Their proof applies also to PCR_k, and we give a sketch that.

Lower bounds on size of PCR_k and RES_k refutations are obtained by the use of a Switching Lemma due to Segerlind et al. [50]. Here we will use that, along with an adapted version for our algebraic proof systems. Such new version is proved in this chapter. In some cases we will rely directly on the results in [50], when the necessary modifications to their proof are trivial and of little interest.

The general lower bound strategy used in this chapter leverages on the relation between two complexity measures: the size measure and another auxiliary measure μ (width for RES_k and degree for PCR_k). Consider a formula F and a refutation Π. The strategy consists in the following steps:

1. Define a distribution on partial assignments $\mathcal{D}$.

2. Show that for any line $l \in \Pi$ and a random assignment $\rho \in \mathcal{D}$, $\mu(l\restriction_\rho)$ is small with very high probability.

3. Show that no refutation Π' exists for $F\restriction_\rho$ such that $\mu(\Pi')$ is small.

If a small Π exists, then by a counting argument there exists a ρ such that $l\restriction_\rho$ is small for every $l \in \Pi$. It means $\mu(\Pi\restriction_\rho)$ is small. This is in contradiction with (3) and proves a lower bound on the size of Π.

The main point of this strategy is that the auxiliary measure is almost unrelated with the actual proof system (we will clarify this notion later) so the third step can be achieved in a simpler proof system than the one under study.

5.1 Pcr$_k$ definition

We introduce the notion of *k-monomial*: an algebraic representation of a k-DNF. As usual we interpret 0 as *true* and 1 as *false* and we translate logical disjunctions as algebraic products. Thus any k-DNF on variables $x_1, x_2, \dots x_n$ can be written as the product of algebraic expressions over $x_1, x_2, \dots x_n, \bar{x}_1, \bar{x}_2, \dots \bar{x}_n$, each of them encoding a logical conjunction of size up to k.

We encode a generic conjunction $l_1 \wedge l_2 \wedge \cdots \wedge l_k$ as the polynomial $1 - \bar{l}_1\bar{l}_2 \cdots \bar{l}_k$. Here l_i is a generic element in $V = \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$. Thus a product of such expressions encodes a k-DNF in a natural way. The empty product is intended to be 1, i.e. the *false* k-DNF. An example of a 3-monomial is: $x_3\bar{x}_2(1 - \bar{x}_5x_2)x_4(1 - x_1\bar{x}_2x_3)$.

Thus k-monomials algebraically represent k-DNFs by the following syntactical transformation

$$\prod_i l_i \cdot \prod_j \left(1 - \prod_{i=1}^{k_j} l_i\right) \longleftrightarrow \bigvee_i l_i \vee \bigvee_j \left(\bigwedge_{i=0}^{k_j} \bar{l}_i\right)$$

Notice that this transformation is a essentially a bijection modulo the fact that a conjunction of one variable x in a k-DNF can be mapped equivalently either to $\bar{x}$ or $(1 - x)$. In our framework the translation does not change the function computed because in our algebraic setting we always consider 0,1 assignments in which for any i the equation $\bar{x}_i = 1 - x_i$ holds.

A line in a Pcr$_k$ proof is a *k-polynomial*, which is a linear combinations of k-monomials. Notice that this is a generalization of the approach of Pcr. In Pcr dual variables are used as placeholders to $(1 - x)$ expression, here we use $(1 - \prod x)$ in a similar way, without using placeholders.

The axioms of Pcr$_k$ includes those of Pcr plus axioms

$$1 - y_1y_2 \cdots y_j - (1 - y_1y_2 \cdots y_j) \quad \text{for } j \le k \text{ and } y_1 \dots y_j \in V$$

which introduce syntactical parentheses and allow to work with k-polynomials.

Analogously, the rules of Pcr$_k$ are those of Pcr with one more rule to deduce k-polynomials

$$\frac{p}{(1 - y_1 \cdots y_j)p} \quad \text{for } j \le k \text{ and } y_1 \dots y_j \in V$$

Notice that the extension of Pcr to Pcr$_k$ is simpler than the one from Res to Res$_k$. We do not need any rule with more than two premises. This allows a simulation of Pcr$_k$ by Pcrwhich is more efficient in term of degree.

A Pcr$_k$ proof of a k-polynomial g from k-polynomials $f_1, \dots, f_n$ (denoted by $f_1, \dots, f_n \vdash_k g$) is a sequence of k-polynomials ended by g, each one obtained from either an axiom or by applying a rule to previously derived k-polynomials. In particular $f_1, \dots, f_n \vdash 1$ is a Pcr$_k$ refutation of $f_1, \dots, f_n$.

5.1.1 Complexity measures for Pcr$_k$

The formulas for k-polynomials are a special case of algebraic circuit (see Section B.2.2). In particular k-polynomials are 3-depth formulas and have a well defined degree complexity measure:

for any k-polynomial p there is a unique multilinear polynomial which computes the same function and which characterizes the value of $deg(p)$ (see [51, 23] or Fact 1.8).

5.2 Relations between Pcr_k and other proof systems

We now study the relation between Pcr_k and related proof systems, more precisely Pc, Pcr and Res_k. For any k-polynomial p we denote as p^* its multilinear representation. Thus we have the following facts about Pc, Pcr, Pcr_k.

Fact 5.1. *In Pcr_k the following holds*

1. *For any k-polynomial p we have that $\emptyset \vdash p - p^*$.*
2. *Pcr_k is complete.*
3. *Any refutation Π of a CNF can be simulated by a refutation Γ in Pc or Pcr such that $\deg(\Gamma) \leq \deg(\Pi) + k$.*

Proof. (1) It is sufficient to prove that the first statement holds for a k-monomials m. We proceed by induction on the number of factors. We denote here a generic factor as $(1 - \prod x)$. If $m = (1 - \prod x)$ then $m - m^*$ is an axiom of Pcr_k. Consider now a k-monomial $(1 - \prod x)m$. By induction $m - m^*$ is derivable, then we get $(1 - \prod x)m - (1 - \prod x)m^*$ by multiplication rule. From axiom $(1 - \prod x) - 1 + \prod x$ we get $(1 - \prod x)m^* - m^* + \prod xm^*$ by applying multiplications and sums. By another sum application we obtain $(1 - \prod x)m - m^* + \prod x\ m^*$. We complete the derivation by applying Pcr boolean axioms to eliminate negated variables and non-multilinear terms from the expanded part. By soundness of Pcr_k and uniqueness of the representation this is the boolean polynomial corresponding to m. The proof can be easily extended to a k-polynomial by acting on all its k-monomial one at time.

(2) $f_1, \ldots, f_n \models g$ implies $f_1^*, \ldots, f_n^* \models g^*$. By completeness of Pc we get $f_1^*, \ldots, f_n^* \vdash g^*$. Finally by using (1) we get $f_1, \ldots, f_n \vdash g$.

(3) Polynomials obtained as a CNF encoding in Pc are proper lines in Pcr_k. Now consider a refutation $\Pi = \{p_i\}_i$ of the encoded CNF. We show how to derive p_i^* in Pc. This is sufficient because $1^* = 1$ thus the result will be a valid refutation. If p_i is a premise then $p_i^* = p_i$. If p_i is an axiom, then either it is an axiom in Pcr or it is a parenthesis axiom. In both cases $p_i^* = 0$. If $p_i = p_a + p_b$ then $p_i^* = p_a^* + p_b^*$. If $p_i = xp_a$ ($p_i = \bar{x}p_a$) then p_i^* is the multilinearization of xp_a^* ($p_a^* - xp_a^*$). If $p_i = (1 - \prod x)p_a$ then the product of $(1 - \prod x)^*$ and p_a^* can be obtained and multilinearized in Pcr. In all such derivations the degree is never higher than $\deg(p_a^*) + k$ which is less or equal than $deg(p_a) + k$. Notice that any proof in Pc is also a proof in Pcr. □

Corollary 5.2. *$\text{Gop}(G)$ can't be proved by Pcr_k in sublinear degree*

The previous lemma tells us that Pcr_k is sound and complete, and even if such proof system can give us some advantage over Pcr in term of proof size, it has essentially the same power in term of degree complexity. Thus the degree lower bound for Pc and Pcr proved in Chapter 3 also holds for Pcr_k. We now show that Pcr_k is (not surprisingly) able to simulate Res_k efficiently in term of size: the number of k-monomials in the Pcr_k translation of a Res_k proof is not much bigger than the number of k-DNF in such proof.

Fact 5.3. *Let Π be a refutation of a CNF ϕ. Let p_ϕ be the set of polynomials arising from the polynomial translation of ϕ. Then there are Pcr_k refutation Γ of p_ϕ such that $S(\Gamma) = O(2^k S(\Pi)^{O(1)})$.*

Proof. We refer to names and notation for Res$_k$ rules given in preliminaries (see Definition 1.1.3). Weakening rule is simulated by multiplication rule. For the other three rules consider the case in which A and B are empty DNFs: by completeness these rules can be easily simulated in size $O(2^k)$ and degree k because they involve at most k original variables. Consider now non-empty k-DNFs A, B and the corresponding k-monomials m_A, m_B. Observe that if $p_1, \cdots p_l \vdash q$ then $m_A p_1, \cdots m_A p_l \vdash m_A q$ in Pcr$_k$ in the same size. Also if $p_1, p_2 \vdash q$ then

$$m_A p_1, m_B p_2 \vdash m_A m_B p_1, m_A m_B p_2 \vdash m_A m_B q$$

in the same size plus the number of factors of m_A and m_B. Now if ϕ is the empty k-DNF then p_ϕ is 1. Thus the simulation is complete. $\square$

5.3 Switching lemma for k-DNF and k-monomials

A standard technique in Computational Complexity is to study how a complexity measure behaves when a computation is randomly restricted. We now study the degree complexity of a k-polynomial under some kinds of restrictions. We also rephrase such study in the framework of Res$_k$ (results in this case are not original and are given without a proof).

Segerlind et al. in [50] reduce lower bounds for Res$_k$ to width lower bounds for Resolution, in a similar fashion we reduce lower bounds on the number of k-monomials in Pcr$_k$ proofs to degree lower bounds for Pc or Pcr. The intuition is that if a random restriction drops the degree complexity of a k-monomial with very high probability, then a small Pcr$_k$ refutation is restricted to small degree with nonzero probability. We will design a random restriction for which the latter event is not possible, which means the Pcr$_k$ refutation is far from being small. The key tool for such line of reasoning is a "switching lemma" which states that the probability of not reducing the degree is very small. We can easily adapt the Switching Lemma in [50]. It studies the restriction of k-DNFs to shallow decision trees. We will prove an analogous Switching Lemma which studies restrictions of k-monomials to multilinear polynomials of low degree.

5.3.1 Decision trees for boolean functions

To introduce the Switching Lemma we need another computational model which represents "computation by cases". The computation is represented by a tree, paths of which correspond to mutually exclusive partial assignments. Paths are augmented until the value of the function is completely determined.

Definition 5.4. *A **decision tree** computes a boolean function $f : \{0,1\}^n \mapsto R$ on n variables $\{x_1 \dots x_n\}$ if it is a tree with the following properties:*

- *leaf vertices are labelled by values in R.*
- *if the root is a leaf labelled by v, then f is a constant function of value c.*
- *internal vertices are labelled by variable names.*
- *if the root has children and is labelled by x_i then it has exactly two children trees T_0 and T_1 where T_0 is a decision tree for $f\restriction_{x_i=0}$ and T_1 is a decision tree for $f\restriction_{x_i=1}$. None of them has a vertex labelled by x_i.*

Despite its simplicity, decision tree is an important computational model with many applications. For our purposes we focus on decision trees for functions over boolean domain. Like for any other computational model we are interested in complexity measures: several decision trees compute the same function so we want a notion of efficiency to rank them.

Definition 5.5. *The **height** of a decision tree T is the length of the longest path from the root to a leaf. We denote it as $\mathrm{ht}(T)$. For a function over boolean domain we denote $\mathrm{ht}(f)$ as the smallest value of $\mathrm{ht}(T)$ for T computing f.*

Our main interest is degree complexity measure: we now relate the height of a decision tree representation with the degree of the polynomial representation.

Fact 5.6. *If $f : \{0,1\} \to R$ is computable by a decision tree of height h then*

1. *If $R = \{true, false\}$ then f can be written as an h-DNF and as an h-CNF.*
2. *If $R = \mathbb{F}$ then $\deg(f) \leq h$.*

Proof. Let be T a tree of height $\mathrm{ht}(T)$ that computes f. Consider any path P from the root to a leaf. The path is identified uniquely by the labels on the internal vertices and the corresponding choices made.

(1) We can easily write a conjunction on $x_1, \ldots x_n$ expressing that an assignment is compatible with the choices on the path. Such conjunctions contain at most h literals. The disjunction of all such expressions corresponding to paths which evaluate to *true* is a h-DNF and is equivalent to f. By negating the leaf labels of T we obtain a decision tree for $\neg f$ of height h, then there is an h-DNF for $\neg f$. By De Morgan Rules we have an h-CNF computing f.

(2) Consider a path from the root to a leaf of T to be $P = (x_{i_1} = v_1, \ldots x_{i_p} = v_p)$. We denote V_0 and V_1 the set of variables in the path assigned respectively to 0 and 1. We denote χ_P as $\prod_{x \in V_0}(1-x) \cdot \prod_{x \in V_1} x$ and f_P the value labelling the leaf on this path. χ_P has degree less than or equal to h, and evaluates to 1 if and only if the assignment is compatible with the path P. The polynomial $\sum_P f_p \chi_P$ computes f. □

Notice that the polynomial representation is unique, though there are many representation as a decision tree of arbitrary height. This means that the transformation of a suboptimal decision tree in a polynomial would cause high degree terms to cancel.

5.3.2 Random restriction

Definition 5.7. *Let ϕ be a k-DNF on $\{x_1, \ldots, x_n\}$. We call $c(\phi)$ the size of the smallest set of variables containing at least one variable from every conjunction in ϕ. We call $c(\phi)$ the **covering number** of ϕ. The covering number of a k-monomial is defined as the same of the corresponding k-DNF.*

Recall Corollary 3.4 in [50]. It says that a random restriction which is **very strong** on sparse k-DNFs (i.e. with high cover number) is also **reasonable strong** on general k-DNFs. "Very strong" means that the restricted k-DNF is a constant with good probability. "Reasonably strong" means the restricted k-DNF has a shallow decision tree with good probability. We also know that degree is lower than the decision tree height. Thus we can correctly rephrase the corollary in our terminology.

Lemma 5.8. *(Corollary 3.4 [50] with $d = 1, \gamma = 1, s = h/2$) Let k be a positive integers, fix a $\delta \in (0,1]$ and let $\mathcal{D}$ be a distribution on partial assignments so that for any k-DNF ψ*

$$\Pr_{\rho \in \mathcal{D}}[\psi \restriction_\rho \neq \mathit{true}] \leq 2^{-\delta c(\psi)}$$

then for every k-DNF ϕ,

$$\Pr_{\rho \in \mathcal{D}}[\mathrm{ht}(\phi \restriction_\rho) > h] \leq k 2^{-h \frac{\delta}{4}^k}$$

Corollary 5.9. *Let k be a positive integers, fix a $\delta \in (0,1]$ and let $\mathcal{D}$ be a distribution on partial assignments so that for any k-monomial f*

$$\Pr_{\rho \in \mathcal{D}}[f\restriction_\rho \neq 0] \leq 2^{-\delta c(f)}$$

then for every k-monomial m,

$$\Pr_{\rho \in \mathcal{D}}[\deg(m\restriction_\rho) > h] \leq k 2^{-h \frac{\delta}{4}^k}$$

Proof. Consider the k-DNF ϕ which computes the same function of m. $\phi\restriction_\rho$ and $m\restriction_\rho$ are the same function, thus we have $\deg(m\restriction_\rho) = \deg(\phi\restriction_\rho) \leq \mathrm{ht}(\phi\restriction_\rho)$ because of Fact 5.6. The assumptions on $\mathcal{D}$ are equivalent to the ones needed by Lemma 5.8, then we can apply it and complete the proof. □

5.3.3 An application: lower bound in PCR$_k$ for weak pigeonhole principle

Several applications of this switching lemma are presented in [50], where exponential lower bounds are given for RES$_k$. We want to remark that in the case such proof strategy applies to PCR$_k$.

Definition 5.10. ***(Pigeon hole principle on a bipartite Graph for*** PCR$_k$***)*** *Fix $G = (U_1, U_2, E)$ a bipartite graph. For any $\{u, v\} \in E$ we define a variable x_{uv}. The principle contains the following polynomials:*

- *Any $u \in U_1$ is in an hole*
$$\prod_{v \in \Gamma(u)} x_{uv}$$
- *Two pigeons are in different holes: for any $\{u, v\}, \{u', v\} \in E$ with $u \neq u'$*
$$\bar{x}_{uv} \cdot \bar{x}_{u'v}$$

Theorem 5.11. *For any $c > 1$ there is an ϵ such that the pigeonhole principle on graph $K_{cn,n}$ has no proofs in* PCR$_k$ *of size less then 2^{n^ϵ}.*

We just give a sketch of the proof. We suggest to check [50] for all details.

Proof. **(Sketch)**

(1) For any $c > 1$ it is considered a bipartite regular graph on cn vertices on a side and n on the other, and $O(\log n)$ edges per vertex. The pigeonhole principle defined on this graph subsumes the one on $K_{cn,n}$. A random restriction is defined by the following process: with probability $1/4$ a hole is matched, and if it is matched then a match is uniformly selected among its neighbours. All others pigeons are disconnected from the hole. Notice that there exists an initial graph which is an expander after the restriction with high probability (see [50]).

(2) Given any set of k-DNFs S of size less than 2^{n^ϵ}. With positive probability such restriction (a) restricts all formulas in S to decision tree height less than $\Omega(n)$; (b) the restricted principle is an instance of pigeonhole principle on an expander graph.

(3) Consider the set of all k-monomials in a proof in PCR$_k$ of the pigeon hole principle. Assume $S < 2^{n^\epsilon}$ then a random restriction transforms all monomials in the proof to degree less then $\Omega(n)$. Using Fact 5.1 we know there is a PC proof for the restricted principle of degree less than $\Omega(n) + k$. This is in contradiction with the expansion properties of the restricted graph: we know such principle can't be proved in such low degree this kind of graphs (see [15, 7]). □

5.3.4 A framework

In the following sections we need a distribution on partial assignment which satisfies the hypothesis of Lemma 5.8 and Corollary 5.9.

Definition 5.12. *For a variable set $\mathcal{V}$ we define a distribution of partial assignments with respect to the variables $\mathcal{V}^l = \{v^i : v \in \mathcal{V}, i \in [l]\}$.*

The distribution is defined according to this process: for any variable $v \in \mathcal{V}$ select uniformly and independently $i \in [l]$ and then for all $j \in [l] - \{i\}$ uniformly and independently assign a $\{0, 1\}$ value to v^j.

We denote this distribution as $D_l(\mathcal{V})$, omitting $\mathcal{V}$ if it is clear from the context.

Lemma 5.13. *Let k be given and let ϕ be a k-DNF on variables $\mathcal{V}^{k+1}$. There exists a constant $\delta > 0$, depending only on k, such that*

$$\Pr_{\rho \in \mathcal{D}_{k+1}}[\phi\restriction_\rho \neq true] < 2^{-\delta c(\phi)}$$

Let k be given and let m be a k-monomial on variables $\mathcal{V}^{k+1}$ and their negations. There exists a constant $\delta > 0$, depending only on k, such that

$$\Pr_{\rho \in \mathcal{D}_{k+1}}[m\restriction_\rho \neq 0] < 2^{-\delta c(m)}$$

Proof. We prove the result for a k-DNFs ϕ. The proof is identical for k-monomials. We say a collection of terms is "independent" when for any $v \in \mathcal{V}$, at most one of its term contains variables in $\{v^1, \ldots, v^{k+1}\}$. The greatest independent collection of terms in ϕ has at least $\frac{c(\phi)}{k(k+1)}$ members otherwise we could build a cover smaller than $c(\phi)$. The probability that an assignment sampled according to $\mathcal{D}_{k+1}$ set a term t to true is independent on how the assignment behaves on other terms in the collection. Inside t at most k elements of a given $\{v^1, \ldots, v^{k+1}\}$ set can occur. Then any of them is assigned with at least probability $\frac{1}{2}$. With probability at least $\frac{1}{4}$ any variable of t is assigned to the value needed to make t true. Then the restriction fails to satisfy the k-DNF with probability at most

$$\left(1 - \frac{1}{4^k}\right)^{\frac{c(m)}{k(k+1)}} < 2^{-\delta c(m)}$$

for a δ which depends only from k. $\square$

5.4 A separation between PCR_k and PCR_{k+1}

In analogy with RES_k, we prove a strict hierarchy result for PCR_k. The main part of the RES_k hierarchy separation in [50] was proving that some contradictions arising from a graph ordering principle are refutable in polynomial size but demand high width in Resolution. In this section we define a variant of $\mathrm{GOP}(G)$, which is polynomially refutable by PCR_{k+1} but it's not polynomially refutable by PCR_k. We follow closely the ideas developed for RES_k in [50].

Let $Even(a_1, \ldots, a_k)$ be the function from $\{0, 1\}^k$ to $\{0, 1\}$ which gives 0 if the number of input variables at 0 are even. Such function is a 2^{k-1} size multilinear polynomial with degree k.

For each variable x_{ab} of $\mathrm{GOP}(G)$ we introduce k new variables $x^1_{ab}, \ldots, x^k_{ab}$. $\mathrm{GOP}^{\oplus k}(G)$ is defined as a modification of $\mathrm{GOP}(G)$: substitute any x_{ab} with $Even(x^1_{ab}, \ldots, x^k_{ab})$. Such principle is specified by kd degree polynomials with less than 2^{dk} monomials each, where d is the degree of of the graph G. The usual axioms apply to the new variables.

Theorem 5.14. *For any graph G, $\mathrm{GOP}^{\oplus k}(G)$ has a polynomial size refutation in PCR_k*

Proof. We give a polynomial PCR_k refutation of an auxiliary principle called $\mathrm{PGOP}^{\oplus k}(G)$, and then we polynomially reduce $\mathrm{GOP}^{\oplus k}(G)$ to $\mathrm{PGOP}^{\oplus k}(G)$.

First notice that $Even(x^1_{ab}, \ldots, x^k_{ab})$ (respectively $1 - Even(x^1_{ab}, \ldots, x^k_{ab})$) can be written as $\prod(1 - l_1 \cdots l_k)$ where $l_1 \cdots l_k$ range among all tuples of variables $x^1_{ab}, \ldots, x^k_{ab}$ with an even (respectively odd) number of negated variables. We denote such k-monomial as $Even_{ab}$ (respectively Odd_{ab}).

$\mathrm{PGOP}^{\oplus k}(G)$ is defined from $\mathrm{GOP}(G)$ as follows: each x_{ab} is substituted with the k-monomial $Even_{a,b}$. $\mathrm{PGOP}^{\oplus k}(G)$ has the property to translate any monomial in $\mathrm{GOP}(G)$ with a single k-monomial in $\mathrm{PGOP}^{\oplus k}(G)$. So a PCR refutation of $\mathrm{GOP}(G)$ can be translated in a PCR_k refutation of $\mathrm{PGOP}^{\oplus k}(G)$ by the mapping

$$\begin{aligned} x_{ab} &\mapsto Even_{ab} \\ \bar{x}_{ab} &\mapsto Odd_{ab} \end{aligned}$$

and the pseudo axioms

$$\begin{aligned} &Even^2_{ab} - Even_{ab} \\ &Odd^2_{ab} - Odd_{ab} \\ &1 - Odd_{a,b} - Even_{a,b} \end{aligned}$$

Each of these pseudo axioms is derivable in PCR_k in size $O(2^k)$. Since $Even_{ab}$ and Odd_{ab} are semantically equivalent to their polynomial expansions, in PCR_k we can derive the polynomials of $\mathrm{PGOP}^{\oplus k}(G)$ from those of $\mathrm{GOP}^{\oplus k}(G)$ with a proof of size at most $2^{O(dk)}$ each. □

We now prove the lower bound for PCR_k. Let $\mathcal{V}$ denote the set of variables of $\mathrm{GOP}(G)$, then $D_{k+1}(\mathcal{V})$ is a distribution on partial assignments of the variables in $\mathrm{GOP}^{\oplus k+1}(G)$.

Notice that when we apply a restriction $\rho \in D_{k+1}(\mathcal{V})$ to $\mathrm{GOP}^{\oplus k+1}(G)$, it reduces to a principle equivalent to $\mathrm{GOP}(G)$. It could happen that some variables have inverted polarity. Anyway it is clear that from a PCR refutation of $\mathrm{GOP}^{\oplus k+1}(G)\restriction_\rho$ we can easily construct a PCR refutation of $\mathrm{GOP}(G)$ of the same degree. Hence applying Theorem 3.4 we have the following Corollary.

Corollary 5.15. *Let G be an (r, c)-vertex expander. Then for any constant $k \geq 1$ and for any $\rho \in D_{k+1}(\mathcal{V})$, there are no PC refutations of $\mathrm{GOP}^{\oplus k+1}(G)\restriction_\rho$ of degree less than or equal to $cr/4$.*

Theorem 5.16. *Let G be $(\delta n, c)$-vertex expander on n vertices, for some $\delta > 1$. Let $k \geq 1$ be a constant. There exists a constant ϵ such that any PCR_k refutation of $\mathrm{GOP}^{\oplus k+1}(G)$ contains at least $2^{\epsilon n}$ k-monomials.*

Proof. Fix $r = \delta n$. By Lemma 5.13 we can apply the Switching Lemma to $\mathcal{D}_{k+1}$ with $h = (rc/4 - k)$. We have that for any k-monomial m,

$$\Pr_{\rho \in D_{k+1}}[\deg(m\restriction_\rho) > (rc/4 - k)] \leq k 2^{-O(rc/4-k)}$$

Hence there exists a constant ϵ such that

$$\Pr_{\rho \in D_{k+1}}[\deg(m\restriction_\rho) > (rc/4 - k)] \leq 2^{-\epsilon n}$$

Assume that there is PCR_k refutation of $\mathrm{GOP}^{\oplus k+1}(G)$ of size strictly less than $2^{\epsilon n}$, then by the union bound there is a PCR_k refutation Π of $\mathrm{GOP}^{\oplus k+1}(G)\restriction_\rho$ with $\deg(\Pi) \leq (rc/4 - k)$. Hence by Lemma 5.1 there is a PC refutation of $\mathrm{GOP}^{\oplus k+1}(G)\restriction_\rho$ of degree $\leq rc/4$. This is in contradiction with Corollary 5.15. □

Using a family of vertex expander (see [36] for constructions) together with Theorems 5.14 and 5.16 we get the following exponential separation.

Theorem 5.17. *For $k = O(1)$, there is a family of proposition over m variables separating exponentially* PCR_k *from* PCR_{k+1}*: there are polynomial size refutations in* PCR_{k+1} *and any refutation in* PCR_k *requires size* $2^{\Omega(\sqrt{m})}$.

5.5 A separation between PCR_k and RES_k

We have already shown that PCR_k simulates efficiently RES_k. Here we argue that PCR_k is strictly stronger than RES_k by showing a proposition with no short RES_k refutation.

We build on the known separation between PCR and RES, given by the *matching principle* on a bipartite graph. We consider two different formulations of such principle. In both of them we fix the underlying graph to be a d-regular, bipartite, $(\Omega(n), O(1))$-boundary expander graph $G = ([n], [n-1], E)$ (see Definition B.7 in Appendix B).

Definition 5.18. ***(Matching principle)*** *For any $\{u, v\} \in E$ we define a variable x_{uv}. The matching principle contains the following clauses:*

- *Any $u \in [n]$ is matched*

$$\bigvee_{v\in\Gamma(u)} x_{uv}$$

- *Any $v \in [n-1]$ is matched*

$$\bigvee_{u\in\Gamma(v)} x_{uv}$$

- *Two left vertices are matched to different right vertices: for any $\{u, v\}, \{u', v\} \in E$ with $u \neq u'$*

$$\bar{x}_{uv} \vee \bar{x}_{u'v}$$

- *Two right vertices are matched to different left vertices: for any $\{u, v\}, \{u, v'\} \in E$ with $v \neq v'$*

$$\bar{x}_{uv} \vee \bar{x}_{uv'}$$

We now extend this principle as we did in the previous section by substituting any variable with the exclusive or of a sequence of $k + 1$ variables. We define the functions

$$Odd(v_1, v_2, \ldots, v_{k+1}) = v_1 \oplus v_2 \oplus \ldots \oplus v_{k+1}$$
$$Even(v_1, v_2, \ldots, v_{k+1}) = 1 \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_{k+1}$$

Definition 5.19. *For any $\{u, v\} \in E$ we define $k+1$ variables $x^1_{uv}, \ldots, x^{k+1}_{uv}$. A vertex u is matched to a vertex v when $Odd(x^1_{uv}, \ldots, x^{k+1}_{uv})$ is true. The principle* $\mathrm{MATCH}^{k+1}(G)$ *is represented by the following propositional formulas:*

- *For any $u \in [n]$ there a matching $v \in [n-1]$*

$$\bigvee_{v\in\Gamma(u)} Odd(x^1_{uv}, \ldots, x^{k+1}_{uv})$$

- *For any $v \in [n-1]$ there a matching $u \in [n]$*

$$\bigvee_{u\in\Gamma(v)} Odd(x^1_{uv}, \dots, x^{k+1}_{uv})$$

- *Two left vertices are matched to different right vertices: for any $\{u,v\},\{u',v\} \in E$ with $u \neq u'$*

$$Even(x^1_{uv}, \dots, x^{k+1}_{uv}) \bigvee Even(x^1_{u'v}, \dots, x^{k+1}_{u'v})$$

- *Two right vertices are matched to different left vertices: for any $\{u,v\},\{u,v'\} \in E$ with $v \neq v'$*

$$Even(x^1_{uv}, \dots, x^{k+1}_{uv}) \bigvee Even(x^1_{uv'}, \dots, x^{k+1}_{uv'})$$

The formulas we used to describe $\textsc{Match}^{k+1}(G)$ are not clauses, but all of them contain a constant number of variables and are representable with at most $2^{d(k+1)}$ clauses each. So $\textsc{Match}^{k+1}(G)$ has linear size representation with respect to the number of vertices of G (remember that d and k are constants).

In this section we again adopt the strategy

1. We show an efficient refutation of $\textsc{Match}^{k+1}(G)$ principle in $\textsc{Pcr}_k$
2. A partial assignment distributed according to $\mathcal{D}_{k+1}$ transforms $\textsc{Match}^{k+1}(G)$ in $\textsc{Match}^1(G)$, which is equivalent to the matching principle.
3. Such restriction reduces a k-DNF to a set of clauses of small width with high probability.
4. If a refutation of $\textsc{Match}^{k+1}(G)$ in $\textsc{Res}_k$ is small, then with positive probability the restricted refutation has small width in spite of being a $\textsc{Res}$ refutation of the matching principle, and this is known to be impossible.

Theorem 5.20. *Let $k = O(1)$, there is a polynomial size and constant degree* $\textsc{Pcr}_k$ *refutation of* $\textsc{Match}^{k+1}(G)$.

Proof. For any edge uv we denote as f_{uv} the polynomial over $x^1_{uv} \dots x^{k+1}_{uv}$ which computes the function $Even(x^1_{uv}, \dots, x^{k+1}_{uv})$. Notice that f_{uv} evaluates to 1 if and only if $Odd(x^1_{uv}, \dots, x^{k+1}_{uv})$ is true, because of the interpretation of 0 and 1 as *true* and *false*.

For any vertex u in the left side of the graph we deduce the polynomial $L_a = 1 - \sum_{v\in\Gamma(u)} f_{uv}$, also for any v on the right side we deduce $R_v = 1 - \sum_{u\in\Gamma(v)} f_{uv}$. Such polynomials are implied by the principle. Notice that these deductions require at most $2^{O(d(k+1))}$ steps and degree at most $d(k+1)$ because $\textsc{Pcr}_k$ is complete and such polynomials contains at most $d(k+1)$ variables each. We then deduce

$$\sum_{u\in[n]} L_u - \sum_{v\in[n-1]} R_v$$

Any polynomial f_{uv} corresponding to an edge $\{u,v\}$ appears once in the first sum and once in the second one. Thus all such polynomials cancel out. Consider the constant terms: 1 is summed n times while it is subtracted $n-1$ times, thus the deduced polynomials is 1. This proof has size $n \cdot 2^{O(d(k+1))}$ and degree at most $d(k+1)$. □

Notice that the previous refutation can be achieved in $\textsc{Pc}$. $\textsc{Pc}$ can simulate each step in constant size and constant degree.

Theorem 5.21. *Let $k = O(1)$, any* $\textsc{Res}_k$ *refutation of* $\textsc{Match}^{k+1}(G)$ *requires exponential size.*

Proof. We consider the distribution D_{k+1} on partial assignments on variables in $\textsc{Match}^{k+1}(G)$. We know by Lemma 5.13 that we can apply Switching Lemma (Lemma 5.8) to any $\textsc{Res}_k$ refutation of $\textsc{Match}^{k+1}(G)$ with $h = \delta n$ where δ is a constant we will fix later. We get that for any k-DNF F in the proof,

$$\Pr_{\rho \in D_{k+1}}\left[\mathrm{ht}(F\restriction_\rho) > \frac{\delta n}{k}\right] \leq k2^{-O(\frac{\delta n}{k})}$$

Hence there exists a constant ϵ such that

$$\Pr_{\rho \in D_{k+1}}\left[\mathrm{ht}(F\restriction_\rho) > \frac{\delta n}{k}\right] \leq 2^{-\epsilon\frac{\delta n}{k}}$$

Assume that there is $\textsc{Res}_k$ refutation of $\textsc{Match}^{k+1}(G)$ of size strictly less than $2^{\epsilon\frac{\delta n}{k}}$, then by the union bound there is a $\textsc{Res}_k$ refutation Π of $\textsc{Match}^{k+1}(G)\restriction_\rho$ with $\mathrm{ht}(\Pi) \leq \frac{\delta n}{k}$. We now use the claim

> **Theorem 5.1 in [50]** Let ϕ be a propositional formula represented by clauses of width at most h. If it has a $\textsc{Res}_k$ refutation Π such that $\mathrm{ht}(\Pi) \leq h$ then it also has a refutation of width hk in $\textsc{Res}$.

This implies there is a resolution refutation of width less than δn for the restricted $\textsc{Match}^{k+1}(G)$. It is easy to translate this to a refutation for matching principle. We know δ can't be too small, because of the following result:

> **Theorem B.10, see Appendix B** If $G = (U_1, U_2, E)$ is a bipartite $(\Omega(n), O(1))$-boundary expander, then any $\textsc{Res}$ refutation of the matching principle on G requires width at least $\Omega(n)$.

Then there is a constant value of δ such that no resolution refutation of width δn exists for matching principle on G. Then no $\textsc{Res}_k$ refutation of size $2^{\epsilon\frac{\delta n}{k}} = 2^{-\Omega(n)}$ exists as well for $\textsc{Match}^{k+1}(G)$. ☐

Chapter 6

Random CNF are hard for $\textsc{Pcr}_k$

In this chapter we show that random 3-CNFs are hard to refute in $\textsc{Pcr}_k$. $\textsc{Pcr}_k$ is the most powerful system for which such result is known. Hardness results for 3-CNFs are known for $\textsc{Res}$ [24, 12, 17], $\textsc{Pc}$ and $\textsc{Pcr}$[15], and $\textsc{Res}_k$ [2].

Here we follow the strategy of Alekhnovich [2]. In that paper a partial assignment distribution is shown which acts similarly to the ones described in the Chapter 5: with high probability a k-DNF is restricted to a low degree polynomial, thus there is a positive probability that all lines in a short proof are restricted to low degree polynomials. In the chapter we will always consider the systems $\textsc{Pc}$, $\textsc{Pcr}$ and $\textsc{Pcr}_k$ defined over a field of characteristic different from 2.

6.1 Random 3-CNF, encodings and $\textsc{Pc}$ lower bounds

Let $\phi_{n,\Delta}$ be the random 3-CNF obtained selecting Δn clauses uniformly from the set of all possible 3-clauses over n variables. Following [2], instead of proving a lower bound for $\phi_{n,\Delta}$ refutations, we will prove it for a polynomial encoding of a set of linear equations modules 2, which semantically implies $\phi_{n,\Delta}$. We will always consider linear systems modulo 2.

For each possible formula $\phi_{n,\Delta}$ consider the matrix $A_{\phi_{n,\Delta}}$ defined by

$$A_{\phi_{n,\Delta}}[i,j] = \begin{cases} 1 & \text{if the } i\text{-th clause of } \phi_{n,\Delta} \text{ contains the variable } x_j \\ 0 & \text{otherwise} \end{cases}$$

Let $b_{\phi_{n,\Delta}}$ be the boolean m vector defined by

$$b_{\phi_{n,\Delta}}[i] = (\#\text{ of positive variables in the } i\text{-the clauses}) \bmod 2$$

We consider the system of linear equations $A_{\phi_{n,\Delta}}x = b_{\phi_{n,\Delta}}$.

Given a sistem of linear equations $Ax = b$, we define its *polynomial encoding* $Poly(A, b)$ as follows: for each equation $\ell \in Ax = b$, let f_ℓ is the characteristic function of ℓ that is 0 if and only if the equation is satisfied. Let $\tilde{\ell}$ be the unique multilinear polynomial representing the function f_ℓ. Then $Poly(A, b) = \bigcup_{\ell \in Ax=b} \tilde{\ell}$. Notice that $deg(\tilde{\ell}) = 3$.

Lemma 6.1. *Each* $\textsc{Pcr}_k$ *refutation of* $\phi_{n,\Delta}$ *can be transformed into a* $\textsc{Pcr}_k$ *refutation of* $Poly(A_{\phi_{n,\Delta}}, b_{\phi_{n,\Delta}})$ *with a polynomial increase in the size.*

Proof. Any equation ℓ in $A_{\phi_{n,\Delta}}x = b_{\phi_{n,\Delta}}$ semantically implies the clause C in $\phi_{n,\Delta}$, from which ℓ arose. Then by completeness we have a $\textsc{Pcr}_k$ proof of the polynomial encoding of C from $\tilde{\ell}$. □

The following observation is crucial to find 3-CNF which are hard for PC, PCR, PCR_k refutation systems. Such result is rephrased and used many times (see [17, 22, 15, 7, 2, 4]) and refers to the boundary expansion of a matrix (see Definition B.8).

Fact 6.2. *([24],[7]) For all constant $\Delta > 0$ and for all $c < 1$, let $\phi_{n,\Delta}$ be a random 3-CNF of n variables and Δn clauses. Then with probability $1 - o(1)$ $\phi_{n,\Delta}$ is unsatisfiable and $A_{\phi_{n,\Delta}}$ is a $(\frac{n}{\Delta^{2/(1-c)}}, c)$-boundary expander.*

The reason we consider the boundary expansion of a random 3-CNF (of the corresponding linear system) is the following theorem, saying expanders need high degree to be refuted by PC and PCR.

Theorem 6.3. *(Theorem 3.10 in [4]) Given an unsatisfiable linear system $Ax = b$ where A is an (r, c)-boundary expander, any* PCR *refutation of $Poly(A, b)$ in a field $\mathbb{F}$ with characteristic $\neq 2$ require degree $\geq \frac{rc}{4}$.*

Definitions and results in the next three subsections are essentially taken from [2], sometimes applied to k-monomials instead of k-DNFs.

6.2 How to restrict $Ax = b$ preserving expansion

In the following subsections we will apply restrictions to linear systems $Ax = b$ where A is an expander. In some cases such restrictions could destroy the expansion of the system. Following [2] in this section we develop a tool which extracts a good expander from the restricted system.

Definition 6.4. *Let A be an $m \times n$ matrix and let $r, c > 0$. For a set $J \subseteq [n]$, the relation $\vdash^e_{J,r,c}$ on the set $[m]$ is defined as follows:*

$$I \vdash^e_{J,r,c} I_1 \text{ iff } |I_1| \leq \frac{r}{2} \wedge \left|\partial I_1 - \left(\bigcup_{i\in I}\{j \ : \ A[i,j] = 1\} \cup J\right)\right| < \frac{c}{2}|I_1|$$

Since r, c will be always clear from the context, from now on we will omit them. Let I and J be subsets of the rows and the columns of a matrix A. Consider the following algorithm $Cl^e(A, I, J)$:

```
$R := [m]$
while (there exists $I_1 \subseteq R$ such that $I \vdash^e_J I_1$)
    $I := I \cup I_1$
    $R := R - I_1$
end
output $I$;
```

Define $Cl^e(J) := Cl^e(A, \emptyset, J)$. Two lemmata are immediate from the definition and proved in [2].

Lemma 6.5. *(Lemma 2.4 in [2]) Let A be any boolean $m \times n$ matrix and let $J \subseteq [n]$. Let $I' = Cl^e(J)$ and let $J' = \bigcup_{i\in I'} A_i$. Let $\hat{A}$ be the matrix obtained from A removing the rows in I' and the columns in $J' \cup J$. Either $\hat{A}$ is empty or it is a $(r/2, c/2)$-boundary expander.*

Proof. For any set of row $I \in \hat{A}$, we will denote $\partial_A I$ and $\partial_{\hat{A}} I$ the boundary computed w.r.t. A and $\hat{A}$ respectively. Assume $|I| \leq r/2$. By construction $\partial_A I \subseteq \partial_{\hat{A}} I \cup J \cup J'$. I has no element in common with $Cl^e(J)$, then $|\partial_A I - (J' \cup J)| \geq \frac{c}{2}|I|$. It follows $|\partial_{\hat{A}} I| \geq \frac{c}{2}|I|$. □

It is important to remark that Cl^e does not increase too much the number of columns to remove from A.

Lemma 6.6. *([2, 5]) If A is an (r, c)-boundary expander and $|J| \leq cr/4$, then $|Cl^e(J)| < \frac{2}{c}|J|$.*

Proof. Assume $|Cl^e(J)| \geq \frac{2}{c}|J|$ and consider $I_1 \cdots I_i \cdots I_l$, the inference of $Cl^e(J)$. Wlog we can assume the I_is to be pairwise dijoint. Consider the first step t such that $C = \cup_{i=1}^{t} I_i$ and $|C| \geq \frac{2}{c}|J|$. Since $|C - I_t| < \frac{2}{c}|J| \leq \frac{r}{2}$ and $|I_t| \leq r/2$, then $|C| \leq r$. Thus $|\partial C| \geq c|C|$ by expansion of A. Then $|\partial C - J| \geq c|C| - |J| \geq \frac{c}{2}|C|$. But at any step each I_i add strictly less than $\frac{c}{2}$ elements to $|\partial C - J|$. We have the contraddiction. □

We combine previous lemmas in a useful tool for restricting linear systems while keeping both unsatisfiability and expansion.

Lemma 6.7. *Consider $Ax = b$ be an m equations, n variables unsatisfiable linear system where A is an (r, c)-boundary expander. Let J be a set of columns (i.e. variables of the system) with $|J| \leq \frac{cr}{4}$. Define:*

- *$I' = Cl^e(J)$ and $J' = \bigcup_{i \in I'} \{j : A[i, j] = 1\}$;*
- *$A_{I'}x = b_{I'}$ the linear system containing rows I' from $Ax = b$;*
- *$\hat{A}$ is the matrix A with rows I' and columns $J \cup J'$ removed.*

Then: (1) $A_{I'}x = b_{I'}$ is a satisfiable system on the variables corresponding to columns $J \cup J'$. For any assignment ρ on such variables which satisfies $A_{I'}x = b_{I'}$, we have that: (2) $(Ax = b) \restriction_\rho$ is $\hat{A}x = \hat{b}$ for some $\hat{b}$, (3) $\hat{A}x = \hat{b}$ is unsatisfiable and $\hat{A}$ is and an $(r/2, c/2)$-boundary expander.

Proof. If $A_{I'}x = b_{I'}$ was unsatisfiable, then by gaussian elimination we could obtain a non empty linear combination of rows resulting in $0 = 1$, in the field $\mathbb{F}_2$ such linear combination is a subset H of rows. No variables in ∂H can be eliminated, so ∂H is empty. Since $|J| \leq \frac{cr}{4}$, then by Lemma 6.6 $|I'| \leq \frac{r}{2}$. Thus $|H| \leq \frac{r}{2}$. But then, by the expansion of A, ∂H can't be empty. This is a contradiction.

$(Ax = b) \restriction_\rho$ is $\hat{A}x = \hat{b}$ because assigned columns become constants and satisfied conditions are set to $0 = 0$.

The expansion of $\hat{A}$ is guaranteed by Lemma 6.5. □

6.3 Normal forms

Let us start by recalling that when speaking of k-monomials, a *term* is a either a variable or an expression of the form $(1 - \prod x_i)$. This definition resembles the definition of term for a k-DNF. For a term t, $V(t) := \{i : x_i \text{ appears in } t\}$.

Let us consider another relation on the set of rows of the matrix A.

Definition 6.8. *([7]) Let A be an $m \times n$ matrix and let $r > 0$. For a set $J \subseteq [n]$ (a set of indices of variables) the relation $\vdash_{J,r}$ on the set $[m]$ is defined as follows:*

$$I \vdash_{J,r} I_1 \text{ iff } |I_1| \leq \frac{r}{2} \wedge \partial I_1 \subseteq \left(\bigcup_{i \in I} \{j \ : \ A[i, j] = 1\} \cup J \right)$$

For $J \subseteq [n]$, $Cl(J)$ is the set of all rows that can be inferred from $\emptyset$ via the relation $\vdash^r_J$. For a term t, $Cl(t) := Cl(V(t))$.

The next lemma is proved in [7, 2] and we omit its proof.

Lemma 6.9. *([7, 2]) If $|J| \leq \frac{cr}{2}$, then $|Cl(J)| \leq \frac{|J|}{c}$.*

Let t be a term over variables $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. We identify t with the linear system over $\{x_1, \ldots, x_n\}$ defined by $x_j = \epsilon_j$ for all x_j appearing in t. We fix $\epsilon_j = 1$ for variables appearing positive and $\epsilon_j = 0$ for variables appearing negative. Such system is satisfied iff $t = 0$.

Definition 6.10. *Let A a $m \times n$ matrix which is a (r, c)-boundary expander and let b be a boolean vector of size m. Let t be a term and let $I = Cl(t)$. t is* ***locally consistent*** *with respect to $Ax = b$ if the system $t \wedge A_I x = b_I$ is satisfiable.*

Lemma 6.11. *([2]) Let $Ax = b$ where A is an (r, c)-boundary expander, with $r > \frac{3}{c}$. Term t is locally consistent with $Ax = b$ iff for any subset I of equations with $|I| < r/2$, the system $t \wedge A_I x = b_I$ is satisfiable.*

Proof. Assume that t is locally consistent with A and that there exists a I s.t $|I| < r/2$ and $t \wedge A_I x = b_I$ inconsistent. Then by linear algebra there exist $I' \subseteq I$ and a $V' \subseteq V(t)$, such that $\sum_{i \in I'}(A_i x - b_i) + \sum_{x_j \in V'}(x_j - \epsilon_j) \equiv 1$. Then it must be that $\partial I' \subseteq V(t)$. Thus $I \subseteq Cl(t)$ which is a contradiction with locally consistency of t. The other direction follows since by Lemma 6.9 $Cl(t) < r/2$. □

Corollary 6.12. *Let $Ax = b$ where A is a $m \times n$ boolean matrix which is an (r, c)-boundary expander, with $r > 3/c$. Then for any set $I \subseteq [m]$ such that $|I| < r/2$ the system $A_I x = b_I$ is satisfiable.*

Proof. The statement follows immediately by proving that the constant 0 is locally consistent with respect to $Ax = b$. This in turn follows since otherwise there would be a set I whose boundary is empty. But this is in contradiction with expansion of A. □

Definition 6.13. *Let A be a boolean $m \times n$ matrix and let b be a boolean m vector. A k-monomial is in* **normal form** *with respect to $Ax = b$ if each of its term is locally consistent wrt $Ax = b$.*

Definition 6.14. *Let $Ax = b$ be an unsatisfiable system where A a is boolean $m \times n$ matrix and b be a boolean m vector. A PCR_k refutation Π of $Poly(A, b)$ is in* **normal form** *with respect to $Ax = b$ if all the locally inconsistent terms wrt to $Ax = b$ appearing in Π are only in monomials of degree $O(k)$.*

We end the section by showing that, as long as $k = O(\log n)$, every PCR_k refutation of $Poly(A, b)$ can be transformed into a PCR_k refutation in normal form with only a polynomial increase in the number of k-monomials.

Lemma 6.15. *Let be a linear system $Ax = b$ where A is an $m \times n$ matrix which is an (r, c)-boundary expander. Let $k = O(\log n)$ and Γ be a PCR_k refutation of $Poly(A, b)$. Then there is refutation Π of $Poly(A, b)$ in normal form and such that $S(\Pi) = S(\Gamma)^{O(1)}$.*

Proof. We first get rid from Γ of the locally inconsistent terms of the form $t = (1 - \prod_{1 \leq i \leq k} x_i)$. We want to replace this term by the constant 1 along the proof. By definition there exists some set $I = Cl(t)$ of rows, with $|I| \leq k/c$, such that t is inconsistent with the system $A_I x = b_I$. By completeness of PCR there must be a PCR proof Γ_t of $\prod_i x_i$ from $Poly(A_I, b_I)$. Such proof involves at most $O(k)$ variables so $S(\Gamma_t) = 2^{O(k)}$ and $deg(\Gamma_t) = O(k)$.

Let Π' be the proof where all occurrences of t will be deleted as follows: t could have been introduced in some k-monomial either by the multiplication rule, in which case in the Π' we simply skip this rule, or it was introduced by some axiom of the form $1 - \prod_i x_i - (1 - \prod_i x_i)$. In this case in the new proof we replace this axiom with the PCR proof Γ_t of $\prod_i x_i$. Notice that the PCR proofs Γ_t could introduce in Π' locally inconsistent terms but only occurring in monomials of degree $O(k)$.

Now we obtain Π getting rid from Π' of the locally inconsistent terms $t = x$ with only one variable. Using the PCR proofs Γ_t of $\bar{x}$, we can delete x in the polynomials of $Poly(A, b)$, in the axioms $1-x-\bar{x}$ and in the axioms x^2-x. A PCR_k axiom containing x can be replaced by a similar axiom without x. So x disappears from Π'. As above the Γ_t PCR proofs are of size $S(\Gamma_t) = 2^{O(k)}$ and degree $deg(\Gamma_t) = O(k)$ and may introduce locally inconsistent terms in Π which only occur in monomials of degree $O(k)$. So Π is in normal form and, since $k = O(\log n)$, $S(\Pi)$ is polynomial in $S(\Gamma)$. □

6.4 Random restriction

In this section we define the distribution $\mathcal{D}$ over partial assignments over $\{x_1 \dots, x_n\}$ that will guarantee the applicability of the Switching Lemma (Lemma 5.9). The distribution is that defined by Alekhnovich in [2].

Definition 6.16. *Let A be a $m \times n$ boolean matrix which is a (r, c)-boundary expander. Let $b \in \{0,1\}^m$. Let X be the set of variables $\{x_1, \dots, x_n\}$. Let $\mathcal{D}_{A,b}$ be the distribution over partial assignments ρ over X obtained by the following experiment: choose a random subset X_1 of X of size $cr/4$. Let $\hat{I} = Cl^e(X_1)$. Let $\hat{X} = X_1 \cup Y_1$, where $Y_1 = \{j : \exists i \in \hat{I} : A[i,j] = 1\}$. ρ is obtained by selecting uniformly at random an assignment $\hat{x}$ for the set of variables whose indices are in $\hat{X}$ that satisfies the system $A_{\hat{I}}\hat{x} = b_{\hat{I}}$.*

The proof of the next main lemma is the same as that of the analogous Theorem 3.1 in [2] where instead of k-DNF we use k-monomials. We use the concept of covering number introduced in Definition 5.7.

Lemma 6.17. *([2]) Let A be a $m \times n$ boolean matrix which is a (r, c)-boundary expander such that A has at most $\hat{\Delta}$ ones in each column. Let $b \in \{0,1\}^m$ and assume $r = \Omega(n/\hat{\Delta})$. For any k-monomial f in normal form,*

$$\Pr_{\rho \in \mathcal{D}_{A,b}}[f \restriction_\rho \neq 0] < (1 - 2^{-k})^{c(f)/\hat{\Delta}^{O(k)}}$$

Corollary 6.18. *There exists a constant D such that, under the assumptions of the previous lemma, for any k-monomial in normal form f we have:*

$$\Pr_{\rho \in \mathcal{D}_{A,b}}[f \restriction_\rho \neq 0] < 2^{-c(f)/\hat{\Delta}^{Dk}}$$

6.5 Main result

We are ready to give the main result of this section.

Theorem 6.19. *For any constant Δ let $\phi_{n,\Delta}$ be a random 3-CNF on n variables and Δn clauses. For $k = o(\sqrt{\log n / \log\log n})$ any refutation of $\phi_{n,\Delta}$ in PCR_k over a field with characteristic different from 2, has size $S > 2^{n^{1-o(1)}}$ with high probability.*

Proof. Assume that $\phi_{n,\Delta}$ is an unsatisfiable formula and $A_{\phi_{n,\Delta}}$ is an (r, c)-boundary expander for some constant $c < 1$ and any $r = \Omega(n)$. Consider the system $A_{\phi_{n,\Delta}}x = b_{\phi_{n,\Delta}}$ as defined in Section 6.1. For easiness of notation let us omit the indices $\phi_{n,\Delta}$ from both A and b. Remember k is $O(\log n)$ and let Γ be a PCR_k refutation of $\phi_{n,\Delta}$ of size S. Then by Lemma 6.1 there is a PCR_k refutation Π of $Poly(A, b)$ of size $S^{O(1)}$.

To apply the Switching Lemma (Lemma 5.9), according to Corollary 6.18 we need to transform the proof Π of $Poly(A,b)$ in a proof of $Poly(\hat{A},\hat{b})$ where k-monomials are in normal form and $\hat{A}$ only contains a constant number $\hat{\Delta}$ of ones in each column.

Pick in A the set J of the $cr/4$ columns with the biggest number of ones, By Lemma 6.7 there is a restriction α that, applied to $Ax=b$, restricts this system to $\hat{A}x=\hat{b}$, where $\hat{A}$ is a submatrix of A with at least the columns J removed and is an $(r/2,c/2)$-expander. Notice moreover that in each column of $\hat{A}$ there are at most $\hat{\Delta} \le \frac{12\Delta n}{cr}$ ones, which is a constant since $r=\Omega(n)$. If we now apply Lemma 6.15 to $\Pi\restriction_\alpha$ we get a PCR_k normal form refutation $\hat{\Pi}$ of $Poly(\hat{A},\hat{b})$ of size at most $S^{O(1)}$.

Let now consider ρ sampled from $D_{\hat{A},\hat{b}}$ according to Definition 6.16 and denote by $A'x=b'$ and Π' respectively the system and the refutation obtained restricting $\hat{A}x=\hat{b}$ and $\hat{\Pi}$ by ρ.

By Corollary 6.18 and by setting the parameter of Lemma 5.9 as follows: $\delta=(1/\hat{\Delta})^{Dk}$ and $h=(rc/64)-k-1$, we have that for any k-monomial in normal form m in $\hat{\Pi}$

$$\Pr_\rho[\deg(m\restriction_\rho) > (rc/64)-k-1] \le 2^{\frac{-rc}{2^{O(k^2)}}}$$

With probability greater than $1-S^{O(1)}\cdot 2^{\frac{-rc}{2^{O(k^2)}}}$ we have that $\Pi'=\hat{\Pi}\restriction_\rho$ has degree complexity strictly less than $(rc/64)-k$ by union bound[1], and it is a refutation of $Poly(A',b')$.

Fix any $c<1$ and $r=\frac{n}{\Delta^{2/(1-c)}}$. Notice that $\rho\in D_{\hat{A},\hat{b}}$ is defined in such a way that Lemma 6.7 applies. Thus A' is an $(r/4,c/4)$-boundary expander. If $S<2^{\frac{rc}{2^{O(k^2)}}}$ then using Lemma 6.1 on Π' we get a PCR refutation of $Poly(A',b')$ of degree less than $\frac{rc}{64}$. This is impossible because of Theorem 6.3, and then it follows $S\ge 2^{\frac{rc}{2^{O(k^2)}}}$.

Since by Fact 6.2 with high probability A is an (r,c)-boundary expander, then the theorem follows. □

Acknowledgment: We would like to thank Nathan Segerlind for interesting and helpful discussions about the paper of M. Alekhnovich [2] and to suggest a new strategy in part of the proof of Lemma 2.4 of [2] that led us to the proof of Lemma 6.15.

[1]Notice that locally inconsistent terms which were not eliminated from $\hat{\Pi}$ occur in monomials of degree at most $O(k)$ because of Lemma 6.15

Chapter 7

Open problems

Our work leaves many interesting unsolved problems.

Simplify/Derandomize non-automatizability: It has already been argued that the proof of non automatizability for Resolution [8] suffers of a complex probabilistic construction. Our proof in Chapter 4 uses the same scheme, and suffers of the same problems. It would be nice to find a simpler proof, even better a deterministic construction. Being this result conditioned to a computational complexity assumption, this could be achieved under a stronger (but still believable) assumption.

A non-automatizability proof is essentially the construction of a formula from an search problem P. Small solutions of the problem map to short proofs for the formula, thus any algorithm giving a short proof also gives hints about the target. Recent connections between computational learning theory and automatization could help in finding hard problems which are appropriate for the reduction, based on the hardness of learning concepts.

Quasi-automatizability: It is conjectured that proof systems like Resolution, $\textsc{Pc}$, $\textsc{Pcr}$ are not quasi-automatizable. The proof used in Chapter 4 cannot be improved to get this result, because it works for treelike Resolution which is quasi-automatizable. The existence of propositions like $\textsc{Gop}(G)$ is a small step toward the solution. We know that some proof search algorithms are very slow because they scan a huge space of possible proof steps even if there is a short refutation. The size of the space is much bigger than a quasi-polynomial of the size of the shortest refutation. The problem is that $\textsc{Gop}(G)$ is very easy to refute. We would need a similar principle for which the short proof is hard to compute. This is similar to ask for a decision problem in ($\mathbf{NP} \cap \mathbf{coNP}$) which is not solvable in quasi polynomial time.

An important fact is that a negative result about quasi-automatizability requires a reduction to a problem which is very hard to approximate. Min-Coloring and Max-Clique are good examples. Unfortunately a statement like “there's no clique of size k”, being NP-hard, cannot be encoded efficiently as a CNF unless **coNP=NP**.

Weak algebraic proof systems: A proof system is said to be weak when the lines in a proof are circuits for which known lower bound exists. Lines in $\textsc{Pcr}_k$ are very peculiar circuits. It would be nice to define stronger proof systems based on more complex and natural kinds of algebraic circuits, while being able to prove lower bounds. Natural examples are algebraic circuits which are sum of products of linear forms, in which either the top of the bottom gates have bounded constant fan-in.

Fine tuning the optimality of degree vs size trade-off: In Chapter 3 we claim that the Graph Ordering Principle implies that degree vs size trade-off is optimal for Polynomial Calculus.

It actually can be improved further by showing a tautology of size n with a proof of size $n^{O(1)}$ and which requires degree $\sqrt{n \log(n)}$.

Bibliography

[1] Agrawal, M., Kayal, N., and Saxena, N. Primes is in p. *Ann. of Math 2* (2002), 781–793.

[2] Alekhnovich, M. Lower bounds for k-dnf resolution on random 3-cnfs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing* (2005), pp. 251–256.

[3] Alekhnovich, M., Ben-Sasson, E., Razborov, A. A., and Wigderson, A. Space complexity in propositional calculus. *SIAM J. Comput. 31*, 4 (2002), 1184–1211.

[4] Alekhnovich, M., Ben-Sasson, E., Razborov, A. A., and Wigderson, A. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput. 34*, 1 (2004), 67–88.

[5] Alekhnovich, M., Hirsch, E. A., and Itsykson, D. Exponential lower bounds for the running time of dpll algorithms on satisfiable formulas. In *31st International Colloquium on Automata, Languages and Programming* (2004), pp. 84–96.

[6] Alekhnovich, M., Johannsen, J., Pitassi, T., and Urquhart, A. An exponential separation between regular and general resolution. *Theory of Computing 3*, 1 (2007), 81–102.

[7] Alekhnovich, M., and Razborov, A. A. Lower bounds for polynomial calculus: Nonbinomial case. In *42nd Annual Symposium on Foundations of Computer Science* (2001), pp. 190–199.

[8] Alekhnovich, M., and Razborov, A. A. Resolution is not automatizable unless W[P] is tractable. *SIAM J. Comput. 38*, 4 (2008), 1347–1363.

[9] Alon, N. Tools from higher algebra. In *Handbook of combinatorics (vol. 2).* MIT Press, Cambridge, MA, USA, 1995, pp. 1749–1783.

[10] Arora, S., and Barak, B. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[11] Beame, P. Proof complexity. In *Computational Complexity Theory* (2004), vol. 10 of *IAS/-Park City mathematics series*, American Mathematical Society, pp. 199–246.

[12] Beame, P., Karp, R. M., Pitassi, T., and Saks, M. E. On the complexity of unsatisfiability proofs for random k-cnf formulas. In *STOC* (1998), pp. 561–571.

[13] Beame, P., and Pitassi, T. Simplified and improved resolution lower bounds. In *37th Annual Symposium on Foundations of Computer Science* (1996), IEEE, pp. 274–282.

[14] Ben-sasson, E. Expansion in proof complexity, phd thesis. Tech. rep., Hebrew University, 2001.

[15] Ben-Sasson, E., and Impagliazzo, R. Random cnf's are hard for the polynomial calculus. In *40th Annual Symposium on Foundations of Computer Science* (1999), pp. 415–421.

[16] Ben-Sasson, E., Impagliazzo, R., and Wigderson, A. Near optimal separation of tree-like and general resolution. *Combinatorica 24*, 4 (2004), 585–603.

[17] Ben-Sasson, E., and Wigderson, A. Short proofs are narrow - resolution made simple. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing* (1999), pp. 517–526.

[18] Bonet, M. L., and Galesi, N. A study of proof search algorithms for resolution and polynomial calculus. In *40th Annual Symposium on Foundations of Computer Science* (1999), pp. 422–432.

[19] Bonet, M. L., and Galesi, N. Optimality of size-width tradeoffs for resolution. *Computational Complexity 10*, 4 (2001), 261–276.

[20] Bonet, M. L., and Galesi, N. Degree complexity for a modified pigeonhole principle. *Arch. Math. Log. 42*, 5 (2003), 403–414.

[21] Bonet, M. L., Pitassi, T., and Raz, R. On interpolation and automatization for frege systems. *SIAM J. Comput. 29*, 6 (2000), 1939–1967.

[22] Buss, S. R., Grigoriev, D., Impagliazzo, R., and Pitassi, T. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci. 62*, 2 (2001), 267–289.

[23] Buss, S. R., Impagliazzo, R., Krajícek, J., Pudlák, P., Razborov, A. A., and Sgall, J. Proof complexity in algebraic systems and bounded depth frege systems with modular counting. *Computational Complexity 6*, 3 (1997), 256–298.

[24] Chvátal, V., and Szemerédi, E. Many hard examples for resolution. *J. ACM 35*, 4 (1988), 759–768.

[25] Clegg, M., Edmonds, J., and Impagliazzo, R. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (1996), pp. 174–183.

[26] Cook, S. A. The complexity of theorem proving procedures. In *STOC* (1971), pp. 151–158.

[27] Cook, S. A., Robert, and Reckhow, A. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic 44* (1979), 36–50.

[28] Cox, D., Little, J., and O'Shea, D. *Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd edition.* Springer, 2007.

[29] Downey, R., and Fellows, M. *Parameterized Complexity.* Springer-Verlag, 1999.

[30] Edmonds, J. Path, trees, and flowers. *Canad. J. Math 17* (1965), 449–467.

[31] Galesi, N., and Lauria, M. Degree lower bounds for a graph ordering principle. Submitted. See `http://www.dsi.uniroma1.it/~galesi/publications.html`.

[32] Galesi, N., and Lauria, M. On automatizability of algebraic proof systems. Submitted. See `http://www.dsi.uniroma1.it/~galesi/publications.html`.

[33] Galesi, N., and Lauria, M. Extending polynomial calculus to k-dnf resolution. *Electronic Colloquium on Computational Complexity (ECCC)*, 041 (2007).

[34] Haken, A. The intractability of resolution. *Theor. Comput. Sci. 39* (1985), 297–308.

[35] Hartmanis, J., and Stearns, R. E. On the computational complexity of algorithms. *Transactions of the American Mathematical Society 117* (1965), 285–306.

[36] Hoory, S., Linial, N., and Wigderson, A. Expander graphs and their applications. *Bull. Amer. Math. Soc. 43*, 4 (2006), 439–561.

[37] Impagliazzo, R., Pudlák, P., and Sgall, J. Lower bounds for the polynomial calculus and the gröbner basis algorithm. *Computational Complexity 8*, 2 (1999), 127–144.

[38] Iwama, K. Complexity of finding short resolution proofs. In *MFCS* (1997), I. Prívara and P. Ruzicka, Eds., vol. 1295 of *Lecture Notes in Computer Science*, Springer, pp. 309–318.

[39] Jukna, S. *Extremal Combinatorics: with Applications in Computer Science.* Springer, 2001.

[40] Krajícek, J. On the weak pigeonhole principle. *Fundamenta Mathematicae 170*, 1-3 (2001), 123–140.

[41] Krajícek, J., and Pudlák, P. Propositional proof systems, the consistency of first order theories and the complexity of computations. *J. Symb. Log. 54*, 3 (1989), 1063–1079.

[42] Lang, S. *Algebra*, 3rd ed. Springer-Verlag, 2005.

[43] Levin, L. Universal sequential search problems. *PINFTRANS: Problems of Information Transmission (translated from Problem Peredachi Informatsii (russian) 9* (1973).

[44] Papadimitriou, C. H. *Computational Complexity.* Addison-Wesley, 1994.

[45] Pudlák, P. On reducibility and symmetry of disjoint np-pairs. *Theoretical Computer Science 295* (2003), 626–638.

[46] Pudlák, P., and Sgall, J. Algebraic models of computation and interpolation for algebraic proof systems. *DIMACS series in Theoretical Computer Science 39* (1998), 279–296.

[47] Razborov, A. A. Lower bounds for the polynomial calculus. *Computational Complexity 7*, 4 (1998), 291–324.

[48] Razborov, A. A. Proof complexity of pigeonhole principles. In *In Developments in language theory (Vienna* (2002), Springer-Verlag, pp. 100–116.

[49] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2002.

[50] Segerlind, N., Buss, S. R., and Impagliazzo, R. A switching lemma for small restrictions and lower bounds for k-dnf resolution. *SIAM J. Comput. 33*, 5 (2004), 1171–1200.

[51] Smolensky, R. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing* (1987), pp. 77–82.

[52] Stalmark, G. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica 33* (1996), 277–280.

[53] Tseitin, G. S. On the complexity of derivations in the propositional calculus, 1968.

[54] van Lint, J. H. *Introduction to Coding Theory (Graduate Texts in Mathematics)*, 3rd ed. Springer-Verlag, 1998.

Appendix A

Notions from Commutative Algebra

In the thesis we study several algebraic proof systems. For both their definitions and analysis we require knowledge of some concepts from commutative algebra. We give here the definitions we are going to use in the rest of the thesis. They are given just for reference, for a deeper understanding we suggest to read a book on the topic, like [28, 42].

A.1 Rings

The basic algebraic structure we use is the ring.

Definition A.1. *A* ***Ring*** *is a set R equipped with two operations $+, \cdot$ for which the following properties hold*

Zero $0 \in R$ *such that* $a + 0 = 0 + a = a$ *for every* $a \in R$

+Comm $a + b = b + a$ *for every* $a, b \in R$

+Inv *For every* $a \in R$ *there is a unique* $(-a) \in R$ *such that* $a + (-a) = 0$

+Assoc $(a + b) + c = a + (b + c)$ *for every* $a, b, c \in R$

·Assoc $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ *for every* $a, b, c \in R$

·DistR $(a + b) \cdot c = a \cdot c + b \cdot c$ *for every* $a, b, c \in R$

·DistL $c \cdot (a + b) = c \cdot a + c \cdot b$ *for every* $a, b, c \in R$

Examples of rings are: integers, polynomials over one or more variables, formal power series, functions with values defined on a ring, square matrices with real values. In each case listed we refer to the customary sum and product operations. With the exception of square matrices, in the previous examples the product operation is also commutative.

Definition A.2. *A ring is called* ***commutative*** *if also the following property holds*

·Comm $0 \in R$ *such that* $a \cdot b = b \cdot a$ *for every* $a, b \in R$

Definition A.3. *A ring is called* ***Ring with unity*** *if also the following property holds*

One $1 \in R$ *such that* $a \cdot 1 = 1 \cdot a = a$ *for every* $a \in R$

Example of a ring without unity is the set of all even integers. In this thesis we mostly consider *commutative rings with unity*, with the exception of *ideals* which in general are rings without unity (we introduce ideals in the next section).

Definition A.4. *A **Subring** of a ring R is a set $R' \subseteq R$ such that if $a, b \in R'$ then $-a \in R'$, $a + b \in R'$ and $a \cdot b \in R'$.*

For any given integer m, the set of integers which are multiples of m is a subring of the integers.

Definition A.5. *A **Field** is a set $\mathbb{F}$ equipped with two operations $+, \cdot$ for which the following properties hold*

RING *$\mathbb{F}$ equipped with $+, \cdot$ is a **commutative ring with unity element**.*

$\cdot$INV *For every $a \in \mathbb{F}/\{0\}$ there is $b \in \mathbb{F}$ such that $a \cdot b = 1$.*

Notice that the multiplicative inverse is always unique, so we can denote it either as $\frac{1}{a}$ or a^{-1}.

Polynomial rings: we consider a field $\mathbb{F}$ and variables $x_1, \ldots, x_n$. The ring of polynomials denoted as $\mathbb{F}[x_1, \ldots, x_n]$ can be defined as follow: it is a commutative ring, all finite products of formal variables is in the ring are in the ring, any linear combination of those is in the ring. Nothing else is. Notice that this is the smallest commutative ring containing $\mathbb{F} \cup \{x_1 \ldots x_n\}$ in which no equation holds other than the ones which are necessary.

In general any polynomial can be written as

$$\sum_i \alpha_i m_i$$

where $\alpha_i \in \mathbb{F}$ and m_is are distinct products of formal variables. Notice that the empty product is 1.

A.2 Ideals, Ring Homomorphisms, Quotient rings

Definition A.6. *Consider a subset I of a ring R. I is said to be an **ideal** if*

- $0 \in I$
- *$a, b \in I$ implies $a + b \in I$*
- *$a \in I$ and $r \in R$ imply $r \cdot a \in I$*

The simplest example of ideal is the set containing only 0. Another common example is the set of ring elements which are multiple of a fixed element r. Notice that: any ideal is also a subring; the only ideal containing the element 1 is the ring itself; the only ideals of a field are $\{0\}$ and the field itself.

Definition A.7. *Consider a subset $G \subseteq R$. The set*

$$I = \left\{ \sum_{g \in G} h_g \cdot g | h_g \in R \right\}$$

*is the **ideal generated** by G.*

It is known that any ideal of the polynomial ring is finitely generated, and that any integer ideal of monovariate polynomials ideal can be generated by a single element. Ideals are strongly related with the notion of homomorphism in a ring.

Definition A.8. *Given two rings R, R', any function $f : R \to R'$ is a **ring homomorphism** if*

- $f(0) = 0$
- $f(a + b) = f(a) + f(b)$
- $f(ab) = f(a)f(b)$

*The map f is said to be a **ring isomorphism** if it is an invertible one to one map and both f and f^{-1} are ring homomorphisms.*

An example of ring homomorphism over the integer is the remainder of division by fixed value.

Definition A.9. *Let R be a commutative ring and I an ideal of R. We denote the equivalence relation $\equiv_I$ as follows: for $a, b \in R$ we have $a \equiv_I b$ when $a-b \in I$. We denote $a+I = b \in R : b \equiv_I a$ the equivalence class containing a. The **quotient ring** R/I is the ring of the equivalence classes with respect to the following operations:*

- $(a + I)(b + I) = ab + I$
- $(a + I) + (b + I) = a + b + I$

It is a standard exercise to prove that the definition is independent from the choice of equivalence classes representation. The next theorem explains the relation between quotient rings and ring homomorphisms. It is called the *first isomorphism theorem* for rings.

Theorem A.10. *(1) Let R be a ring and f a ring homomorphism from R to R'. Let I be the kernel of f and $\Im f$ the image of f.*

- *I is an ideal of R*
- *$\Im f$ is a subring of R'*
- *$\Im f$ and R/I are isomorphic*

(2) Let R be a ring and I an ideal of R. The map $a \mapsto a + I$ is a ring homomorphism with kernel I and image R/I.

The next theorem tells that quotient ring constructions behave well when composed. This is customary called *third isomorphism theorem* for rings. Beware the second and third isomorphism theorems are ofter inverted in literature.

Theorem A.11. *Let R be a ring and I, J two ideals in R such that $I \subseteq J$, then*

- *I is an ideal of J*
- *J/I is and ideal of R/I*
- *$(R/I)/(J/I)$ is isomorphic to R/J*

Definition A.12. *Let R be a ring and I be and ideal, then the **residue** of $a \in R$ modulo I is a canonical element of $a + I$. We denote it as $R_I(a)$.*

The previous definition depends from the choice of the canonical element. Such choice is arbitrary and usually depends from the context.

A.3 Quotient ring representations of boolean function

In the thesis we ofter manage polynomials, polynomial ideals and quotient ring. We always assume the formal variables to be evaluated on a boolean domain. It is often convenient to represent this framework as quotient in polynomial rings.

$\mathcal{P}$**:** The algebraic structure of boolean functions evaluating in $\mathbb{F}$ is the ring of multivariate polynomials over $x_1 \dots x_n$ modulo the ideal generated by $x_i^2 - x_i$ for i from 1 to n. We name this ring $\mathcal{P}$.

Fact A.13. *The following holds for* $\mathcal{P}$

- *Given* $r \in \mathcal{P}$*, any representation of* r *computes the same boolean function.*
- *Elements of* $\mathcal{P}$ *have unique representation as multilinear polynomials in* $\mathbb{F}[x_1, \dots, x_n]$.

Proof. By definition of quotient ring any two representations of an element $r \in \mathcal{P}$ differ by a sum of multiples of $x_i^2 - x_i$. Thus the difference evaluates to zero for any boolean assignment. This means that the boolean function computed is completely specified by r. Denote as I the ideal generated by $\{x_i^2 - x_i\}_{i\in[n]}$. Consider a monomial $x^e m$ where m does not contain any occurrence of x and $e \geq 2$. Then $(x^2 - x)x^{e-2}m$ is in the ideal I, so $x^e m \equiv_I x^{e-1}m$. By repeating until $e = 1$ we get xm. Doing the same for all variables we get a multilinear monomial. Thus for any polynomial which represents an element in the ring, we can find a multilinear one representing the same element. Uniqueness holds because ring elements define uniquely boolean functions, and two different multilinear representations define two different functions by Fact 1.8. □

$\mathcal{N}$**:** As in the previous case such expressions have a polynomial quotient ring characterization: $\mathbb{F}[x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n]$ modulo the ideal generated by the polynomial sets $\{x_i^2 - x_i\}_{i\in[n]}$ and $\{1 - x_i - \bar{x}_i\}_{i\in[n]}$. We denote this ring as $\mathcal{N}$.

Fact A.14. *The following holds for* $\mathcal{N}$

- *Given* $r \in \mathcal{N}$*, any representation of* r *computes the same function.*
- *Elements of* $\mathcal{N}$ *have unique representation as multilinear polynomials in* $\mathbb{F}[x_1, \dots, x_n]$.

Proof. The proof is identical to the one of $\mathcal{P}$. Any boolean assignment puts to zero all generators of the quotient ideal. Then any two polynomial representations of the same ring element evaluate identically. This gives the first part.

For the second part: notice that negated variables can be eliminated from a polynomial representation of the ring element r by using the $\bar{x}_i = 1 - x_i$ equations. Then we can apply Fact A.13 to obtain a multilinear polynomial representing r.

Uniqueness holds because of Fact 1.8. □

The discussion about $\mathcal{P}$ and $\mathcal{N}$ clarifies what kind of objects are the proof lines in Polynomial Calculus and Polynomial Calculus with Resolution (See Section 1.1.2 in Chapter 1).

Appendix B

Notions from Computational Complexity

In this thesis we describe several concepts from computational complexity which are useful for the development of the thesis results. We give basic definitions and properties.

B.1 Complexity classes

Definition B.1. *A* decision problem *or a* language *on an alphabet Σ is a subset of Σ^* (i.e. the set of all strings on the alphabet Σ).*

A Turing machine M accepts an input x if the machine M with input x halts in an accepting state. We say $M(x)$ is *true* if M accepts x. We say $M(x)$ is *false* if M halts in a rejecting state. We say $M(x)$ is undefined when M does not halt on input x.

Definition B.2. *We say M decides a language $L \subseteq \Sigma^*$ when its behaviour on Σ^* follows the conditions:*

$x \in L$ then $M(x) = true$

$x \notin L$ then $M(x) = false$

In computational complexity theory it is customary to group languages according to the minimal resources needed by an accepting computation. Such computation is modelled as either a Turing Machine, a combinatorial circuit or some variants of them (e.g. alternating turning machines, decision trees, branching programs,...). For a treatise about this fascinating theory we suggest [44, 10].

Definition B.3. *A set of languages grouped in this way is called a* complexity class.

We now show some basic and important complexity classes. In the following we denote the length of a word x as $|x|$.

P: a language L is said to be in the class **P** if there exists a machine M which halts on any input x after at most $p(|x|)$ steps, where p is a polynomial. We say M runs in *polynomial time*.

NP: a language L is said to be in the class **NP** if there exists a machine M, a polynomial p such that for all $x \in L$, there exists a y such that $M(x, y) = true$; for all $x \notin L$ and any y we require $M(x, y) = false$. We require M to halt in at most $p(|x|)$ steps in both cases.

coNP: a language L is said to be in the class **coNP** if $\Sigma^* - L$ is in **NP**.

The common understanding about **P** is that it collects the decision problems with efficient sequential algorithms, like deciding if a graph has a perfect matching [30] or if a number is a prime [1]. **NP**expresses languages for which there is an easy verifiable witness for language membership. For example deciding if a CNF is satisfiable may seem hard, but a satisfying assignment can be easily checked. In a dual fashion it is easy to check that proposition is not a tautology when given a counterexample. Then the set of proposition tautologies is the complement of a language in **NP**. Then it is a language in **coNP**.

It is not known if whether **P**=**NP**or not. This is *the major question* in computational complexity. Another important question has been discussed in this thesis: is **NP**= **coNP**? This is the founding question for modern proof complexity, because it translates in "do all propositional tautologies have a short proofs?".

Algorithm designers soon discovered that randomization helps computation. Eventually randomized computation complexity classes were defined: **RP** and **coRP** are worth a mention. In this case we consider randomized Turing machines, in which the transition function at each step is a random variable.
RP: a language L is said to be in the class **RP** if there exists a randomized machine M which runs in polynomial time and such that for all $x \in L$, M accept x with probability at least $1/2$. If $x \notin L$ then M never accept x.

RP is similar to **NP**, but x membership requires a huge number of witnesses: half of the whole witness space. This implies that a random generated witness proves $x \in L$ with non trivial probability. In this interpretation of randomized computation the witness encodes the sequence of transitions made by the machine.

coRP: a language L is said to be in the class **coRP** if $\Sigma^* - L$ is in **RP**.

B.1.1 Reduction and NP-completeness

One of the first tools devised in theory of computation was the concept of reduction. Assume any instance of a problem A can be transformed in an instance a problem B and the solution of the latter can be transformed back in a solution of the former. An algorithm for B implies an algorithm for A. Then the impossibility of solving A implies that B cannot be solved either. This concept is used, sometimes implicitly, in almost every impossibility result. It was proved by Cook [26] an Levin [43] that any instance of a decision problem L in **NP** can be transformed in a CNF such that the instance is in the language L if and only if the CNF is satisfiable. This transformation can be done in polynomial time, then **P**= **NP** if and only if CNF satisfiability can be solved in polynomial time. A problem with this property is called **NP**-complete.

B.2 Combinatorial circuits

We define the computational model of circuits. Particular instances of such model will be used. In particular we are interested in those algebraic circuits called "polynomials" and in those boolean circuits called DNF and CNF.

Given a set of n variables in a domain $\mathcal{D}$, a combinatorial circuit on such variables is a direct acyclic graph, in which any vertex is equipped with an ordering on its incoming edges. A vertex in the circuit is called *gate*. The incoming edges of a gate are called *inputs* of the gate and the outgoing edges are called *outputs* of the gate.

Gates with no outputs are called *output gates*, and gates with no inputs are called *input gates*.

An input gate is labelled with a function defined of $\mathcal{D}^n$ with values in $\mathcal{D}'$ (not necessarily equal to $\mathcal{D}$). Most of the time those functions turns out to be projections on one of the parameters. Any other gate is labelled by a function from $\mathcal{D}'^k$ to $\mathcal{D}'$ where k is equal to the number of its inputs.

Evaluation of a circuit Given values to the variables we assign values to the gates by induction on the topological order to the vertices in the graph.

Fix $v_1 \dots v_n$ to be values in $\mathcal{D}^n$, then an input gates labelled the function f evaluates to $f(v_1, \dots, v_n)$.

Consider any other gate that: it is labelled by a function f; it has incoming edges from l gates, each evaluating to values $d_1, \dots, d_l$; then we say the gate evaluates to $f(d_1, \dots, d_l)$ and we call l the *arity* of the gate.

A circuit with a single output gate is said to evaluate to the value of its output gate. Thus a combinatorial circuit *computes* the function which maps elements of $\mathcal{D}^n$ to values in $\mathcal{D}'$ assigned by the evaluation. In the following we will identify a circuit with the function it computes unless it is necessary to refer the particular computation device. The context will clarify the actual meaning.

B.2.1 Complexity measures on circuit

Given a function F on a given domain, we are interested in knowing what is the best circuit computing that function. Such notion of best may depends on several conflicting complexity measures (i.e. the circuit with smallest size, or with smallest depth). Even worse our definition doesn't prevent cheating, like defining the output gate to computes the needed function. Any reasonable computational model requires some constraint on the computing power of gates. In this framework questions about circuit complexity make sense. Circuit complexity is the understanding of how small the following measures can be for a circuit which computes a given function.

Size The size of a circuit is the number of gates in the circuit.

Formula size If all but the input and output gates have out degree equal to one then the combinatorial circuit is called *formula*. It is interesting to study the size of the smallest formula.

Depth We say input gates are at *level* 0 in the circuit. Any other gate's level is one plus the maximum among the levels of the incoming neighbors. The depth of a circuit is the maximum level among the outputs gates. An alternative definition is that the depth is the length of the longest directed path in the circuit.

B.2.2 Algebraic circuits

Consider a combinatorial circuit in which the domain $\mathcal{D}$ is a field $\mathbb{F}$, input gates computes only projection functions, and functions corresponding to gates are: products and linear combinations of elements in $\mathbb{F}$. Such circuit is called an algebraic circuit. Notice that we don't allow division but this is not an issue here, and it saves us to consider ill-defined functions.

A *polynomial* is a circuit (actually a formula) which computes a linear combination of products of variables. It is straightforward to design a depth 2 algebraic circuit for it.

In general the input of an algebraic circuit is the underlying field, but in our cases we are mostly interested in $\mathcal{D} = 0, 1$.

B.2.3 Boolean circuits

Boolean circuits are defined on n variables with values in the domain $\{\mathit{true}, \mathit{false}\}$ of boolean logic. Input gates computes either x or $\neg x$ for some variable x, and the other gates compute either logical conjunction (AND) or logical disjunction (OR). Boolean circuits are usually allowed to have NOT function as internal gates. In this case we avoid it, nevertheless that is irrelevant for computational complexity investigations.

Among boolean circuits some are worth a remark. *Clauses* (resp. *Terms*) are depth 1 circuit constituting respectively a disjunction (resp. conjunction) of input gates. A CNF is a conjunction of clauses. A DNF is a disjunction of terms. We call k-CNF a CNF in which all clauses have at most arity k, similarly we call a k-DNF a DNF in which all terms have at most arity k. Notice that clauses, terms, CNFs and DNFs are formulas.

B.3 Parameterized Computational Complexity

Several optimization problems (i.e. finding the best combinatorial object according to a cost measure) are computationally difficult. There is no algorithm known to solve them efficiently, and several results in computational complexity suggest no such algorithm exists at all.

Several approaches has been developed to overcome this problem: one is that of approximation algorithms, in which the solution asked for is acceptable if its cost is close to the optimum. Another approach is the one of considering problems where the size of the solution is small compared to the input.

Consider the problem of vertex cover: to find the smallest set of vertices in a graph such that all other vertices are at distance one from them. The corresponding decision problem is deciding if there is a vertex cover of size at most k. This problem can be easily solved by listing all sets of k vertices to find one which is a cover. If there are n vertices, there are $O(n^k)$ cases to check. There exists indeed an algorithm which runs in $O(n^2 + 1.274^k)$ [29]. Assuming that k is small compared to n, then the running time is not too tragic.

Parameterized Complexity is a theory designed to model problems in which the size of the solution is small, and a running time which is exponential in small parameter is acceptable.
FPT: A parameterized decision problem on alphabet Σ is $L \subseteq \Sigma^* \times \mathbb{N}$. Such language L is *fixed parameter tractable* if there is an algorithm which decide if $(x, k) \in L$ with a running time of $f(k) \cdot |x|^{O(1)}$. Notice that f can be any function. By definition f is recursive.
FPR: It is similar to **RP**. A parameterized decision problem L is in **FPR**if there exists a randomized Turing machine which runs in time $f(k) \cdot |x|^{O(1)}$ on any input (x, k), accepts with constant probability when the input is in L and rejects with probability 1 otherwise.
coFPR: It is the complement of **FPR**.

An important problem from Parameterized Complexity Theory is **Minimum Monotone Circuit Satisfying Assignment**. It is an optimization problem: given a monotone circuit it asks for the smallest hamming weight of an assignment which makes the circuit to output 1. An instance for the parameterized version of the problem is (C, k) where C is a combinatorial circuit made of gates $\vee, \wedge$ and k is a natural number. Language MMCSA is the set containing all pairs (C, k) where C has a satisfying assignment of Hamming weight less than or equal to k. Now consider a parameterized problem L. We say that L is *fixed parameter reducible* to MMCSA if there is an algorithm A such that for any (x, k) gives $A(x, k) = (C_x, k')$ where:

- (C_x, k') is in $MMCSA$ if and only if $(x, k) \in L$.
- A runs in $f(k) \cdot |x|^{O(1)}$ for some f.
- $|C_x| \leq |x|^{O(1)}$.
- $k' \leq g(k)$ for some arbitrarily g.

W[P]: it is the complexity class of all problems which are fixed parameter reducible to MMCSA. Notice that if MMCSA is fixed parameter tractable then all members of **W[P]** are. Then MMCSA is said to be **W[P]**-complete with respect to fixed parameter reduction. The major theoretical question in Parameterized Complexity is to find out if **FPT**=**W[P]**. This is the parameterized complexity equivalent of the **P** versus **NP** problem.

B.4 Expanders

We use different kinds of expanders in the thesis. Definitions in literature may have some nontrivial differences from the following ones. Such differences would only be technical.

More details about the use of expansion in proof complexity can be found in [24, 14]. For a general tutorial about expander graphs we suggest the survey [36].

Definition B.4. *Let $G = (V, E)$ be a graph and X, Y two disjoint subsets of V. We denote the following sets as*

- $E(X, Y) = \{\{x, y\} \in E : x \in X, y \in Y\}$ *the* ***edge cut*** *between X and Y.*
- $\Gamma(X) = \{y \in V : y \in V/X$ *and* $\{x, y\} \in E\}$ *is the* ***neighborhood*** *of X.*
- $\partial X = \{y \in V : y \in V/X$ *such that* $|E(X, \{y\})| = 1\}$ *is the* ***boundary*** *of X.*

Definition B.5. *Given a graph $G = (V, E)$, we say G is an (r, c)-**edge-expander** if for all $S \subseteq V$ with $|S| \leq r$ we get $|E(S, \bar{S})| \geq (1+c)|S|$. We say G is an edge-expander if it is an $(\Omega(n), \Omega(1))$-edge expander.*

Definition B.6. *Given a graph $G = (V, E)$, we say G is an (r, c)-**vertex-expander** if for all $S \subseteq V$ with $|S| \leq r$ we get $|\Gamma S| \geq c|S|$. We say G is a vertex-expander if it is an $(\Omega(n), \Omega(1))$-vertex expander.*

Definition B.7. *Given a bipartite graph $G = (V, E)$, we say G is an (r, c)-**boundary expander** if for all $S \subseteq V$ with $|S| \leq r$ we get $|\partial S| \geq c|S|$. We say G is a boundary expander if it is an $(\Omega(n), \Omega(1))$-boundary expander.*

We also use the notion of expansion of a matrix which is essentially a generalization of the boundary expansion of a graph.

Definition B.8. *([4, 7, 2]) Let A be a $m \times n$ matrix. For a set of rows I we define the **boundary** of I (denoted as ∂I) as the set of all $j \in [n]$ (the boundary elements) such that there exists exactly one index $i \in I$ for which M_{ij} is non zero. Then, A is a (r, c)-**boundary expander** if the following condition holds: for all $I \subseteq [m]$, if $|I| \leq r$, then $|\partial I| \geq c \cdot |I|$. We say A is a boundary expander if it is an $(\Omega(n), \Omega(1))$-boundary expander.*

Definition B.9. *Let ϕ be a CNF with m clauses and n variables, and M_ϕ be an $m \times n$ matrix defined as follows*

$$M_{ij} = \begin{cases} 1 & \textit{the } j\textit{-th variable occurs in the } i\textit{-th clause of } \phi \\ 0 & \textit{otherwise} \end{cases}$$

*If M_ϕ in an (r, c)-**boundary expander** then so it is ϕ.*

Expansion plays a fundamental role in proof complexity lower bounds. We give an example here, which has been useful in Chapter 5. The following proof uses a technique which is standard since Ben-Sasson and Wigderson seminal work [17] on expansion in proof complexity.

Theorem B.10. *If $G = (U_1, U_2, E)$ is a bipartite $(\Omega(n), O(1))$-boundary expander, then any* Res *refutation of the matching principle on G requires width at least $\Omega(n)$.*

Proof. Assume that G is an (r, c)-boundary expander and fix $V := U_1 \cup U_2$. We refer to Definition 5.18 in Chapter 5 for the matching principle. For any $v \in V$ we denote $P_v := \bigvee_{u \in \Gamma(v)} x_{uv}$, and we denote $\mathcal{B}$ the set containing all other clauses. For any clause C in a resolution refutation of the principle we define $\mu(C)$ as the size of the minimum set $S \subseteq V$ such that

$$\mathcal{B} \cup \{P_v\}_{v \in S} \models C$$

For $P \in \mathcal{B}$, $\mu(P) = 0$. For $v \in V$, $\mu(P_v) = 1$. For any clause C deduced by C_1, C_2, $\mu(C) \leq \mu(C_1) + \mu(C_2)$. Furthermore $\mu(\emptyset) > r$ because the boundary expansion guarantees that any r vertices are matchable. We show that there is a clause C^* in the refutation with $\frac{r}{2} \leq \mu(C^*) \leq r$: search among the proof clauses, starting from the empty clause and going up to a premise if its value of μ is bigger then r. The process stops to a clause with measure greater that r whose both premises have measure less than r. At least one of them must have measure bigger than $\frac{r}{2}$ by the sublinearity of μ.

Now pick a minimal $S \subseteq V$ such that $\mathcal{B} \cup \{P_u\}_{u \in S} \models C^*$.

We claim that for any $u \in \partial S$ there is $x_{uv'} \in C^*$ for some $v' \in V$. This conclude the proof because it means the width of $C^* \geq \frac{|\partial S|}{2} \geq \frac{c|S|}{2} = \Omega(r) = \Omega(n)$.

Assume $u \in \partial S$. There is a unique $v \in S$ which is a neighbor of u. By minimality assumption on S there is an assignment α which satisfies $\mathcal{B}$ (i.e. it is a partial matching) and $\{P_x\}_{x \in S - \{v\}}$ (i.e. it matches all vertices in S but v) but does not satisfies either P_v or C^*. If α doesn't match u to anything we extend it to match v. Otherwise α matches u to some $v' \in V/S$, and we change it to match u to v. We call the new assignment α'. In both cases α and α' differ only by variables indexed by v. Notice that α' satisfies $\mathcal{B} \cup \{P_x\}_{x \in S}$ and then C^*. This means C^* has variables indexed by v. $\square$