On the Complexity of Propositional Proof Systems

Nicola Galesi

e-mail: galesi@ lsi.upc.es

Advisor: Prof. Maria Luisa Bonet

A thesis submitted in conformity with the requirements

for the PhD degree

Departament de Llenguatges i Sistemes Informàtics

Universitat Politècnica de Catalunya

Dedication

This work is dedicated to my father Vittorio (1939 - 1977) and to my grandfather Nicola (1906-1983). The memory of my father has always helped me to go forward. My grandfather taught me the beauty of spending a night trying to solve problems, and the perfect armony of a good solution.

Acknowledgments

I have always thought that the moment in which I had started to write the Acknowledgment, would have meant the end of this adventure. Sometime I have dreamed of this moment. This moment is finally here. I have really finished.

First of all I have to express my deepest gratitude to my Advisor, Prof. Maria Luisa Bonet for all she has done for me. I think this has no price. It was a great, great luck for me that she joined this Department while I was in the first year of the PhD and accepted me as her student. On a professional level I think that she has taught how to really do research, how to analyze the problems and how try to solve them. Possibly she doesn't imagine that on a more personal level she has taught me a lot of things, only by looking to her behaviors. I think that I have been really lucky in finding a person like Maria Luisa. Thanks a lot.

I would like to thank all the people of the Theoretical Computer Science Section of my department for the very stimulating environment in which I have lived during these years. I would express my sincere thanks in particular to Prof. Jose Luis Balcázar for the encouragement he always gave me and for his help in trying to find funds for me and to Prof. Conrado Martínez for being my advisor in the Marie Curie fellowship and for helping me in a lot of occasions. Also I would thank Prof. Jacobo Torán for introducing me in the field of Proof Complexity.

Among the people of the section I want to give a particular thanks, to another PhD student of the Department: Juan Luis Esteban. During these years we have not only shared research, but we have someway helped each other in those moments in which all it seemed not to work. Really thanks Juan Luis !

I would also thank the Secretaries of the Department for their super-efficiency in everything and for being always very kindly with me. Thanks to Judit Cardona, Rita García, Ana Ibáñez, Mercè Juan and Núria Sánchez.

My first year in the Department LSI was supported by a one year grant of the University of Bari. The European Community granted me with a Marie Curie Training a Mobility of Researchers grant under the project "A Structural Complexity Approach to Propositional Proof Systems" for three years. If I'm finishing the PhD in a great part is due to these funds I received. Thanks.

I also take this occasion to thanks all the people that were PhD students with me. In particular I miss a lot Amalia Duch and Miguel Carillo who shared the room with me for several years.

I also would thank three person from University of Bari: Prof. Salvatore Caporaso from the Department of Computer Science. He introduced me in the field of Theoretical Computer Science and he took me with him to work as Research Assistant in the department. Prof. Vito Michele Abrusci form the Department of Philosophy who gave me the opportunity to learn Linear Logic. Prof Mario Coclite from the department of Mathematics also played an important role in my decision to do research.

From Italy I would thank the following person for existing: Nico Nannavecchia, Enzo Checchia, Katia D'Aprile, Enzo Tarantino.

Finally I thank my family: Claudia, Laurenzia and Paolo, for all the incredible support that unconditionally they always have given to me during these years in which I have been far from my city and our house. I have miss them very much and more than any other thing.

For the very end I have leaved the most important person, my companion: Marina Simonetti. She has wait for me and with me during these years. Her continuous support has been one of the fundamental reasons I have arrived at the end.

Abstract

In the thesis we have investigated the complexity of proofs in several propositional proof systems. Our main motivation has been to contribute to the line of research started by Cook and Reckhow to obtain how much knowledge as possible about the complexity of different proof systems to show that there is no super proof system. We also have been motivated by more applied questions concerning the automatic generation of Theorems. The results presented in the thesis were obtained in the papers [15, 14, 16]

Regarding the complexity of proofs we prove both lower and upper bounds for the size of the proofs in several proof systems (resolution and some of its restrictions, Cutting Planes, Polynomial Calculus, Frege systems). Our results give better or new separations between such proof systems. On the other hand our work also concerns with automated theorem proving questions in resolution and Polynomial Calculus. Some of our results imply that restricting the search space to seek for resolution refutations is not always a good strategy. We show that a recently proposed algorithm that finds resolution proofs cannot have a good performance. We also compare it with the Grobner basis algorithm used to find proofs in Polynomial Calculus, a proof systems based on polynomials.

Contents

Chapter 1

Introduction

1.1 The Importance of Proof Complexity

The main motivation for studying the complexity of proofs in logical systems comes from Complexity Theory.

Let P be the class of the decision problems solvable in deterministic polynomial time, let NP be the non deterministic version of P, and *co-NP* the complement of NP. Obtaining the exact relationships among these three classes is considered one of the main goals in Complexity Theory.

An important problem in Logic is to compute the length of the shortest proof of a given tautology in some proof system. Cook and Reckhow in [31] gave a close relationship between this problem and the $NP \neq$ co-NP question. They showed that $NP \neq$ co-NP if and only if *for every propositional proof system there exists a class of tautologies $\mathcal{T}_n$, such that the size of the shortest proof of any $\mathcal{T}_n$ is not bound by any polynomial in the size of $\mathcal{T}_n$.* So, they proposed the following line of research to attack the $NP \neq$ co-NP problem: prove superpolynomial lower bounds for the length of proofs in propositional systems of increasing complexity, in order to acquire sufficient knowledge to obtain the result for every proof system. This line of research has been one of the basic motivations to study lower bounds for the length of proofs in propositional proof systems.

The work program has developped and it has created a new branch of complexity theory, which at present is perfectly developed and established. Since the work of Cook and Reckhow many results have been obtained along the lines of their program. Nevertheless many problems remain open and the fundamental question of whether $NP \neq$co-NP has yet to be solved. Moreover the development of such a program has also pointed out a strict relation between other branches of complexity theory. The theory of complexity of boolean functions has provided us with techniques (and models) that have been successfully applied to obtain important and difficult results regarding the complexity of several propositional proof systems. This relation is not yet fully understood and is a subject *in fieri*, that could bring new results in the areas of both complexity of proof systems and complexity of boolean functions. Finally, the importance of *proof complexity* can be well resumed by the following words of A. Razborov, A. Yao and A. Wigderson quoted from [70]: [..] *We propose a more systematic study of natural reducibilities between such systems: this would help convincing combinatorists and complexity theorists (and ourselves) that proof complexity is a little bit more than just the Hilbert-style game with abstract symbols on a sheet of paper.*

The importance of proof complexity also comes from more applied areas of Computer Science. In fact, from a more applied point of view, studying the size of proofs in logical systems is also important for the field of the automatic generation of theorems. Questions like: how fast we can produce a proof of a tautology in some proof system, are closely related with the complexity of proofs. The algorithms mainly employed in implementing automatic theorem prover procedures are essentially based on the Resolution system or on its restrictions. Nevertheless, in the case of these systems (and also of others), there are renown results proving that the shortest refutation of certain formulas is exponentially long in the size of the formula itself. Therefore, for such systems we cannot hope to produce proofs of some formulas in polynomial time. On the other hand, it could be very useful to have a polynomial time algorithm producing proofs of those formulas that have polynomial size proofs.

From a theoretical point of view, a natural definition capturing the previous property is the following (see [18]): a proof system P is *automatizable* if there is an algorithm $\mathcal{A}_P$, such that for every tautology F, $\mathcal{A}_P$ finds a proof of F in the systems P in time polynomial in the shortest proof of F in P.

Recently (see [6, 10]) it turned out that some techniques used to obtain lower bounds for Resolution, give insight to develop Resolution-based proof-search algorithms. However, at present, it is not known whether Resolution is automatizable or not. Therefore an interesting and promising problem in proof complexity is to study the automatizability of Resolution and other systems we still do not know whether they are automatizable or not. Results showing the no automatizability of systems stronger than resolution are known, though restricted to be conditional to some strongly believed cryptographic conjectures.

1.2 Contributions and Organization of the thesis

In the thesis we show results about both the complexity and the automatizability of proof systems. Regarding the former problem, we prove both lower and upper bounds for the size of the proofs in several proof systems (resolution and some of its restrictions, Cutting Planes, Polynomial Calculus, Frege systems). Our results give better or new separations between proof systems. On the other hand our work has been also motivated by automated theorem proving questions in resolution and Polynomial Calculus. Some of our results imply that restricting the search space to seek for resolution refutations is not always a good strategy. We show that a recently proposed algorithm (see [10]) that finds resolution proofs cannot have a good performance. We also compare it with the Grobner basis algorithm used to find Polynomial Calculus proofs.

The thesis is structured in four chapters. In Chapter 2 we give preliminary definitions of proof systems and we discuss some of the fundamental results about the complexity of proofs. In Chapter 3 we show results about the complexity of proofs in Hilbert propositional calculi and its extensions. In Chapter 4 we mainly show lower bounds for Resolution and Cutting Planes, improving the known separations between these two systems. This Chapter also includes the result extending the separation of the monotone NC hierarchy to the case of real circuits. In Chapter 5 we consider the recently obtained result about the size-width trade-off for Resolution (see [10]). We prove the optimality of this trade-off in the case of the unrestricted Resolution. We also derive a degree lower bound for a modified version of the Pigeon Hole Principle in Polynomial Calculus.

In the rest of this section we explain in more details the contribution this thesis brings to proof complexity.

1.2.1 Hilbert style propositional calculi

Frege systems correspond to Hilbert style calculi for propositional logic. Very little is known about lower bounds for tautologies in Frege systems. In spite of the simplicity of propositional proof systems such as the Hilbert Calculus or the Gentzen sequent Calculus, we are admitedly far at present from proving that these systems are not polynomially bounded. Surprisingly, one of the main difficulties is the lack of families of tautologies candidate to be hard for these systems. M.L. Bonet, S. Buss and T. Pitassi in [13] defined classes of tautologies associated with some difficult combinatorial principles and showed how to prove them in Frege with at most quasipolynomial size proofs. Moreover, contrary to a Cook's conjecture, S. Buss in [21] showed that the Pigeon Hole Principle tautology (used to obtain exponential lower bounds for other weaker propositional proof systems like resolution [41, 6] or Bounded-depth Frege [7, 61]) can be proved in Frege with polynomial size proofs. Finding exponential lower bounds for these systems is considered a very difficult task.

Therefore one of the ways to approach the problem is to try and obtain polynomial lower bounds. At present the only known results about lower bounds for the Frege system were shown by S. Buss ([22]); he gave a method to prove quadratic lower bounds for the number of symbols and linear lower bounds for the number of lines. Also, P. Pudlak and S. Buss in [65] devised a new method, based upon interactive proofs, to obtain linear lower bounds for tree-like Frege. One of the most important challenges in proof complexity is to improve these results. The lower bounds given by S. Buss also hold for Extended Frege (a generalization of the Frege system in which we consider a rule that allows abbreviations of formulas), but fail in the case of Substitution Frege (another generalization of Frege in which we allow a substitution rule). For this last system the best known result is due to Urquhart, in [80]. He showed how $\Omega(n/\log n)$ lines are needed in Substitution Frege to prove some formulas, defined in [80], generalizing those used by Buss in [22].

In Chapter 3 we reformulate the technique used by Urquhart in terms of the Incompressibility Method of the Kolmogorov Complexity, giving a more general framework to obtain his result. We extend it to the case of tree-like Substitution Frege, obtaining a linear lower bound. We prove that the lower bounds for both tree-like and dag-like Substitution Frege are tight, giving explicit optimal proofs for the tautologies employed for the lower bounds. We extend the work of Urquhart showing that lower bounds similar to the previous ones can also be given for another class of tautologies.

In the same Chapter 3 we also study two more issues about Frege systems. First, we prove that tree-like Substitution Frege polynomially simulates dag-like Substitution Frege. This property was only known for Frege and Extended Frege ([31, 12, 55]). Second, we consider a restriction of the Substitution rule where only a renaming of the variables is allowed (Renaming Frege); we observe that adding this rule to tree-like Frege does not add any power to the system. Hence, in spite of the fact that the tree-like case is as powerful as the dag-like one for Frege, Substitution and Extended, we do not expect that this property also holds for Renaming Frege. This is because using jointly [22] Buss result that dag-like Renaming Frege is as powerful as Substitution Frege, and oru result we would obtin that Frege polynomially simulates Subsitution Frege. This contradicts the widely accepted conjecture that Freg cannnot polynomiallly simulate Substitution Frege.

1.2.2 Resolution, Cutting Planes and Monotone Real Circuits

As we have seen in the previous section, for Frege system and most of its extensions the tree-like versions are polynomially equivalent to the general (dag-like) versions. Usually this property does not hold for all proof systems. We have investigated separations between tree-like and unrestricted versions for the systems of resolution and Cutting Planes.

In Chapter 4 we prove an exponential lower bound for tree-like Cutting Planes of a set of clauses for which we give polynomial size resolution refutations. This result has as a consequence an exponential separation between tree-like and dag-like proofs for both Cutting Planes and resolution; in both cases we improve the previously best known separations that were only superpolynomial[17, 48].

From the sequence of papers [17, 63, 52], lower bounds for the size of monotone circuits are very important to give lower bounds for those proof systems (e.g. resolution and Cutting Planes) for which we can obtain a (monotone) feasible interpolation theorem (see [63] for details). Our exponential lower bound for tree-like Cutting Planes uses the interpolation theorem. We obtain exponential lower bounds in Cutting Planes using the Interpolation technique as a consequence of proving an exponential lower bound for the size of monotone real tree-like circuits computing a given monotone boolean function. This result extends the separation of the monotone NC hierarchy of R. Raz and P. McKenzie [66] to the case of monotone real circuits. As in their work, we prove an $\Omega(n^{\epsilon})$ lower bound on the depth of monotone real circuits computing a monotone boolean function that can be computed by a polynomial size monotone circuit.

A commonly used strategy to find Resolution proofs is to reduce the search space by using restricted versions of Resolution that are still complete. Therefore, an exponential separation between a restriction of resolution and unrestricted resolution, shows that finding restricted proofs cannot always be a good strategy to find unrestricted proofs. A consequence of our previous exponential separation is that finding tree-like resolution proofs can be exponentially slower than finding general resolution proofs.

We consider several restrictions of resolution lying between tree-like and dag-like resolution, such as Regular, Davis-Putnam, Negative and Positive. We are motivated by the fact that these restrictions (e.g. Davis-Putnam) are often used to implement proof search algorithms, and that only superpolynomial separations from general resolution were known (see the work of Goerdt [38, 39, 40]).

On the one hand we investigate separations of tree-like resolution from these restrictions, and on the other we investigate separations of these restrictions from unrestricted resolution. In the former case we obtain an exponential separation of tree-like from regular, Davis-Putnam and negative resolution. These results extend the corresponding previously known superpolynomial separations (see [79]).

In the latter case we prove an exponential separation between Davis-Putnam resolution and unrestricted resolution; the upper-bound proof also respects the restriction of being a negative resolution proof so that the separation is better. Only a superpolynomial separation between Davis-Putnam and unrestricted resolution was previously known.

1.2.3 Optimality of the size width trade-off and degree lower bounds

Resolution was one of the first proof systems for which superpolynomial lower bounds were obtained. Tseitin [] showed that a family of unsatisfiable formulas expressing a property of graphs cannot be proven with a polynomial proof in regular resolution. In 1985 A. Haken [41] obtained the first exponential lower bound for unrestricted resolution using the set of clauses associated to the Pigeon Hole Principle. Several subsequent papers have refined Haken's work, both by extending his result to other families of formulas (Tseitin formulas and Random formulas [77, 28]), and by simplifying his technique. In particular the work of T. Pitassi and P. Beame [6] simplifies Haken's technique enormously by giving Resolution lower bounds for both the Pigeon Hole Principle and for Random Formulas. Recently E. Ben-Sasson and A. Wigderson in [10] considered the size of the largest clause in a refutation (the *width*) as a complexity measure for resolution refutations. They gave a general trade-off between the width and the size of a resolution refutation, reducing the problem of giving lower bounds for the size to that of giving lower bounds for the width. This way they obtained a unified method to prove most of the known lower bounds for resolution. The width-size relationship can be stated as follows: if F, an unsatisfiable formula over n variables, has a resolution refutation of size S, then it has a refutation of width $O(\sqrt{n \log S})$. Through the width-size relationship they devised a new simple proof-search algorithm based on seeking for clauses of increasing size.

The trade-off of [10] shows that for k-CNF, with k a constant, if we can give a width lower bound equal to the number of variables, then we have an exponential lower bound for the size of refuting the formula. An immediate interesting question arising from the work [10] is whether we can improve the size-width trade-off there obtained. That is, can we get a weaker width lower bound (e.g. to the square root of the number of variables), but still obtain an expoential lower bound for the size?

In Chapter 3 we show that the size-width trade-off for unrestricted Resolution is tight and therefore cannot be improved to obtain better lower bounds for the size. We also study whether for some known restrictions of resolution it is possible to improve the trade-off given for unrestricted resolution. We give a negative answer to this question for the following restrictions: Regular, Davis-Putnam, Positive, Negative and Linear. Our result has two main consequences. First, that the proof-search algorithm that [10] propose for Resolution is not going to be efficient and moreover cannot be used to prove the automatizability of resolution (and its restrictions). Second, our result provides exponential separations between tree-like resolution and all the restrictions considered. The separation from Linear resolution is completely new and previously unknown. The others were only recently obtained in [14].

Polynomial Calculus is a refutation system introduced by M.Clegg, J. Edmonds and R. Impagliazzo in [29] working on CNF translated to polynomials. The proof complexity measure for Polynomial Calculus is the maximal degree of the polynomials used in the refutation.

As observed in [10] the width measure plays for resolution the same role the degree measure has for Polynomial Calculus. Hence, given our optimality result, it is natural to ask whether some polynomial formulation of the formula we use requires $\Omega(n)$ degree refutations in Polynomial Calculus. This would show that there is some tautology provable fast in resolution but not in Polynomial Calculus. To obtain this result we should prove that the Polynomial Calculus simulation of resolution given in [29] is optimal when we consider formulas with a polynomial size resolution refutation. A simple observation shows that, in this case, this simulation cannot be better than a quasipolynomial.

Among the tautologies with polynomial size resolution refutations that may be employed to obtain a degree lower bound in Polynomial Calculus there is $MPHP$, a formula encoding a modified Pigeon Hole Principle defined and used by Goerdt in [39]. We study degree lower bounds in Polynomial Calculus for this formual and we derive a $\Omega(\log_2 n)$ degree lower bound, applying the technique recently introduced by Razborov in [69] (and simplified in in [46]) for the Pigeon Hole Principle. Our result shows that this technique also works for a tautology different, even if only slightly, from the Pigpern Hole Principle.

Nevertheless, our degree lower bound is not so good as to obtain the separation desired between Polynomial Calculus and resolution (it simply gives another proof of the above observation). So this remains an interesting open problem. We think that some polynomial formulation of the tautology we use for the trade-off optimality result will require high degree Polynomial Calculus proofs.

Finally we show a new simple Polynomial Calculus simulation of resolution whose main consequence is that under a fixed standard translation of CNF to polynomials the width based algorithm of Ben-Sasson and Wigderson cannot have a better performance in finding proofs than the Grobner Basis proof search algorithm of Clegg et al.

Chapter 2

Preliminary Definitions and Fundamental Results

In this Chapter we introduce most of the concepts used in the thesis, giving formal definitions of the propositional proof systems. We discuss the fundamental and most important results so far obtained for these systems. In the last section of the Chapter, we give a brief introduction to the main notions of circuit complexity we need in the rest of the thesis (especially in Chapter 4). The aim is that of introduce the technique based on the *Interpolation Theorem* to give lower bounds in Resolution and Cutting Planes proof systems. Technique we use in our result [14] discussed in Chapter 4.

2.1 Concrete Propositional Proof Systems

A *propositional formula* is either a boolean variable, or inductively defined from two formulas using the set $\{\wedge, \vee, \neg, \rightarrow\}$ of connectives.

The definition of abstract propositional proof system was formalized by Cook and Reckhow in [31]. Let $TAUT$ be the set of all tautologies expressed as propositional formulas.

Definition 2.1.1 *An* ABSTRACT PROOF SYSTEM *is a surjective function* $f : \{0,1\}^* \rightarrow TAUT$ *which is computable in polynomial time.*

The idea of this definition is that if $f(w) = A$, then w is (an encoding in $\{0,1\}^*$ of) a proof of A. This general definition allows to define as a proof system any logical system. The only requirements are (1) that the system is complete, i.e. any tautology has a proof, and (2) that we are able to check the validity of a proof in the system in polynomial time. Let $|w|$ denote the length of the string $w \in \{0,1\}^*$. Consider the following definition:

Definition 2.1.2 *A proof system f is* SUPER *if every propositional tautology has a polynomial size proof in f.*

Studying the length of proofs in propositional proof systems is closely related to important and long-standing open problems in Complexity Theory. The reason is the following Theorem, whose proof is not hard considered the co-NP completeness of $TAUT$.

Theorem 2.1.1 *([31]) There exists a super proof system if and only if NP = co-NP.*

Since it is conjectured that $NP \neq$ co-NP, then we would like to prove that every propositional proof system is not super. This means find, for every propositional proof system, classes of tautologies whose shorter proofs are exponentially long. Cook and Reckhow proposed to try to obtain this kind of result for proof systems of increasing complexity until having sufficient knoweldge to prove the result for all proof systems. To approach this problem we need a method to compare the efficiency of different propositional proof systems with respect to their ability to prove tautologies using shorter proofs. Cook and Reckhow in [31] defined a notion of *simulation* between proof systems. The idea of this reducibility relation is to preserve upward the property of being a *super* proof system.

Definition 2.1.3 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two proof systems over the same language. $\mathcal{P}$'* POLYNOMIALLY SIMULATES *$\mathcal{P}$, if there is a polynomial p such that for every formula A and for every $\mathcal{P}$-proof π of A, there exists an $\mathcal{P}'$-proof π' of A such that (1) $|\pi'| \leq p(|\pi|)$; and (2) the mapping transforming π in π' is polynomial time computable.*

There are proof systems that are much more powerful than others. The following definition formalizes this concept.

Definition 2.1.4 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two proof systems over the same language. We say that $\mathcal{P}$ is* EXPONENTIALLY SEPARATED *(respectively* SUPERPOLYNOMIALLY SEPARATED*) from $\mathcal{P}'$ if there is a class of tautologies $\{A_n\}_{n\in N}$,where $|A_n| \leq p(n)$ for some polynomial p, such that:*

- *There is an $\mathcal{P}'$-proof π' of A_n and a polynomial p such that $|\pi'| \leq p(n)$.*
- *The shortest proof of A_n in $\mathcal{P}$ is $\geq \Omega(2^n)$ (respectively $\geq \Omega(n^{\log_2 n})$).*

The previous two definitions can also be given when the two proof systems are not defined over the same language, provided there is a polynomial time translation from formulas in the first language to formulas in the other language.

In the next section we introduce concrete examples of proof systems which are subject of the thesis. Let $[n]$ be the set $\{1, 2, \ldots, n\}$.

2.1.1 Resolution and its restrictions

Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be a set of boolean variables. For a variable $x_i \in \mathcal{V}$ let $\bar{x}_i$ denote the negation of x_i. A literal ℓ_i can be either a variable x_i or its negation, with the convention that if $\ell_i = \bar{x}_i$, then $\bar{\ell}_i = x_i$ and that $\bar{\bar{x}}_i = x_i$. A clause is a disjunction of literals $\{\ell_{i_1}, \ldots, \ell_{i_k}\}$, eventually empty. A $C(onjuntive)N(ormal)F(orm)$ formula is a conjunction of clauses. A k-CNF is a CNF formula whose clauses are defined by at most k literals. For a formula F, let $Lit(F)$ be the set of literals of F.

RESOLUTION is a refutation proof system for formulas in CNF form based on the resolution algorithm of [72]. It uses the following *resolution rule.*

$$\frac{C \vee x \quad \bar{x} \vee D}{C \vee D}$$

where if C and D have common literals, they appear only once in $C \vee D$. A RESOLUTION REFUTATION of a CNF formula F is a derivation of the empty clause from the clauses defining F, using the above inference rule. The *length* or *size* of a refutation is the number of clauses.

Resolution was the first proof system for which tautologies require exponentially long proofs were found. In the work [76] Tseitin showed that a family of unsatisfiable formulae expressing a property (which we do not define here) of graphs cannot be proven with a polynomial size refutation in regular resolution, a restriction or resolution we define below. It is remarkable to note that this result was obtained more than 10 years before the work of Cook and Reckhow. Later Z. Galil in [36] refined the work of Tseitin. Only in 1985 A. Haken in [41] obtained the first exponential lower bound for the general resolution system. We present here the formula he used to give his result since it is one of the most important and widely used class of tautologies in complexity of proofs.

The *Pigeon hole principle* says that if $n+1$ pigeons are sitting in n holes then two pigeons are sitting in the same hole. Consider the variables $p_{i,j}$ for $1 \leq i \leq n+1$ and $1 \leq j \leq n$ expressing that the pigeon i is sitting in the hole j. Then propositional formula PHP_n^{n+1} associated to the principle is given by:

$$\bigwedge_{1 \leq i \leq n+1} \bigvee_{1 \leq j \leq n} p_{i,j} \rightarrow \bigvee_{1 \leq i < m \leq n+1} \bigvee_{1 \leq j \leq n} (p_{i,j} \wedge p_{m,j})$$

Haken introduced a new technique showing that the smallest size resolution proof for the negation of PHP_n^{n+1} is exponential in n. This method was very important, and it was used by Buss and Turan in [26] to extend the result of Haken to the case of PHP_n^{cn}, for any constant $c > 1$, and by Urquhart in [77] to give a new exponential lower bound for resolution for a formula obtained applying the Tsetin property to certain expander graphs. V. Chvátal and E. Szemerédi in [28], extended Urquhart's work by showing that almost all unsatisfiable k-CNF formulas require exponential size in Resolution. Beame and Pitassi greatly simplified and improved Haken's method in [6] by obtaining lower bounds both for PHP_n^{n+1} and for random formulas. Pudlak in [63] gave another general method based upon the interpolation property to obtain exponential lower bounds in Resolution using a formula related to a a boolean function (see Section 4 for more details). Very recently Ben-Sasson and Wigderson in [10] using the size of clauses as complexity measure for Resolution proofs (see Chapter 5 for more details), gave a unified method to obtain most of the known lower bounds results for the Resolution system. In the work [16] we prove that the method of [10] is optimal and cannot be used to improve existing lower bounds results.

Due to the fact that resolution is the system most frequently used in Theorem prover applications, several restrictions of resolution system are also well studied. In particular we focus our attention upon the following systems (see [74] for a more complete lists of variants of Resolution)

- TREE-LIKE resolution. Any clause in the proof is used at most once as premise in a resolution rule.

- REGULAR resolution. The proofs are restricted in such a way that any variable can be eliminated at most once in any path from an initial clause to the empty clause.

- DAVIS-PUTNAM resolution (DP-Resolution, for short). The proofs are restricted in such a way that there exists an ordering of the variables such that if a variable x is eliminated before a variable y on any path from an initial clause to the empty clause, then x is before y in the ordering. Notice that Davis-Putnam resolution is necessarily regular.

- NEGATIVE resolution (N-resolution, for short). The application of the resolution rule is restricted to the case in which one of the premises must not contain any positive literal.

- POSITIVE resolution (P-resolution, for short). The symmetric of N-resolution.

- LINEAR resolution. The refutation can be defined by a sequence of clauses $(C_0, C_1, \dots, C_n)$ such that C_0 is an initial clause, C_n is the empty clause and for all i, $1 \leq i \leq n$ in the resolution step $\frac{C_{i-1} \quad B_{i-1}}{C_i}$, the clause B_{i-1} is either an initial clause or such that $B_{i-1} = C_j$ for some $j < i$.

As for resolution for any of the above systems, the *length* or *size* of a refutation, is the number of clauses in the refutation.

One of the main questions about restrictions of resolution is to known whether they are less powerful than general resolution. This questions is also relevant to the field of automatic generation of proofs (see Chapter 4 and 5 for more details). Superpolynomial separations between N-resolution, Davis-Putnam resolution and regular resolution on one side and general resolution on the other were shown by Goerdt in the sequence of papers [38, 39, 40]. In the work [14] we extend the separation for Davis-Putnam from superpolynomial to exponential.

Separations between tree-like and restrictions of Resolution are also investigated. Johannsen in [48] showed a superpolynomial separation between tree-like and unrestricted resolution. We improve this separation to exponential in [14], showing that it is even better since the upper bound proofs also holds for Davis-Putnam (and therefore regular) resolution. In [16] we find a tautology that exponentially separates tree-like resolution from all the considered restrictions.

2.1.2 Cutting Planes

The *Cutting Planes* system, CP for short, is a refutation system for CNF formulas introduced in [32] working on linear inequalities. Starting with a set of linear inequalities corresponding to clauses, we derive a contradiction using a set of rules. Given an unsatisfiable CNF formula F over variables $x_1, \ldots, x_n$ we think of the variables as integers assuming only 0 or 1 values, where 0 means $FALSE$ and 1 $TRUE$. Each clause of F of the form

$$\bigvee_{i=1}^{k} x_{j_i} \vee \bigvee_{i=1}^{m} \neg x_{l_i}$$

is translated into the linear inequality

$$\sum_{i=1}^{k} x_{j_i} + \sum_{i=1}^{m} (1 \perp x_{l_i}) \geq 1$$

In general the CP system works on linear inequalities of the form $\sum_{i=1}^{m} a_i x_i \geq A$ where $a_1, \ldots, a_m$ and A are integer coefficients and $x_1, \ldots x_m$ are 0-1 variables. The CP system has four rules:

1. Basic algebraic simplifications like adding or deleting terms of the form $0x_i$

2. Addition of two inequalities:

$$\frac{\sum_{i=1}^{m} a_i x_i \geq A \qquad \sum_{i=1}^{m} b_i x_i \geq B}{\sum_{i=1}^{m} (a_i + b_i) x_i \geq A + B}$$

3. Multiplication of an inequality by an integer:

$$\frac{\sum_{i=1}^{m} a_i x_i \geq A}{\sum_{i=1}^{m} c a_i x_i \geq cA} \quad c \in Z$$

4. Division of an inequality by an integer:

$$\frac{\sum_{i=1}^{m} a_i x_i \geq A}{\sum_{i=1}^{m} \frac{a_i}{c} x_i \geq \lceil \frac{A}{c} \rceil} \quad c, \frac{a_i}{c} \in Z$$

Since variables are over $\{0, 1\}$ we will add for each variable x the axioms $x \geq 0$ and $\perp x \geq \perp 1$. Let F be an unsatisfiable formula in CNF and let $In(F)$ be the set of linear inequalities associated to the clauses of F. A *CP refutation* of F is a *proof* of the inequality $0 \geq 1$ from the initial inequalities $In(F)$ using the CP rules. In [32] was proved that CP is a sound and complete refutation system for formulas in CNF.
The *length* of a CP proof is the number of linear inequalities used in the proof, and the *size* is the number of binary symbols needed to encode the entire proof. By [32] size and length are polynomially equivalent so it is not important here to distinguish between them. A CP proof will be called *tree-like* if every linear inequality in the proof is used at most once. In a *dag-like* CP proof we can use many times any inequality.

The first results about the complexity of proofs in this system were given in [32]. They defined the system and, giving polynomial size Cutting Plane proofs for the set of inequalities corresponding to $\neg PHP_n^{n+1}$, deduced that the Cutting Planes system is exponentially separated from resolution and therefore more efficient. Pitassi, Impagliazzo and Urquhart in [45] were the first to obtain an exponential lower bound in the case of tree-like Cutting Planes. Then [17] considered the restriction to Cutting Planes proofs in which only polynomially bounded coefficients were allowed in the linear inequalities. For this system they showed an exponential lower bound for an unsatisfiable formula related to a monotone boolean function. Finally Pudlak in [63] and Cook and Haken in [42] gave general circuit complexity results from which exponential lower bounds were obtained even in the Cutting Planes system with unrestricted coefficients. It is important to note that all the cited results concerning Cutting Planes use models and results from the complexity of boolean functions. Finally it is known that the tree-like Cutting Planes proof system is less efficient than the dag-like one (see [17, 48, 14]).

2.1.3 Polynomial Calculus

The Polynomial Calculus (PC) is a refutation system for formulas in CNF introduced in By Clegg, Edmonds and Impagliazzo in [29]. *CNF* formulas F are encoded as a sequence of polynomials $p_1, = 0, \ldots, p_m = 0$ over some field K. To force 0-1 solutions we always add among the initial polynomials the axioms $x^2 \perp x = 0$ for all variable x. A PC refutation is a sequence of polynomials ending with $1 = 0$ such that each line in the sequence is either an initial polynomial or is obtained from two previous polynomials in the sequence by the following rules:

1. sum: $\frac{f \quad g}{\alpha f + \beta g}$ for $\alpha, \beta \in K$;

2. product: $\frac{f}{xf}$, for any variable x.

The degree of a refutation is the maximal degree of a polynomial used in the proof. The complexity of a *PC* refutation is given by its degree.

We define a *standard mapping* tr from formulas in *CNF* to sets of polynomials in the following way: (1) $tr(x) = 1 \perp x$; (2) $tr(\bar{x}) = x$; (3) $tr(x \vee y) = tr(x) \cdot tr(y)$.

The Polynomial Calculus system was defined in [29] and considered in the work [27]. The first work giving a significant degree lower bounds for Polynomial Calculus was that of Razborov [69]. There he showed that a polynomial formulation of the PHP_n^{n+1} requires degree $\Omega(n)$. Several subsequent works, both simplify his technique (Impagliazzo, Pudlak and Sgall in [46]) and extend the result of Razborov, introducing new technique to show degree lower bounds for other tautologies [24], and to almost all unsatisfiable CNF formulas in [9]. All these results shows that polynomial formulations of formulas hard to prove in Resolution require high degree to be refuted in Polynomial Calculus. In the work [16] we approach the problem of finding a formula easy to refute in resolution, but requiring high degree in Polynomial Calculus. Applying a technique similar to that of [69, 46], we show a $\Omega(\log n)$ degree lower bound for a polynomial formulation of a formula having polynomial size resolution refutations (see Chapter 5 for details).

2.1.4 Frege systems and Gentzen Systems

A FREGE PROOF SYSTEM is a propositional language with a finite number of *axiom schemes* and a finite number of *rules of inference*. Any Frege system must be *sound* and *complete*.

A typical example of Frege systems is the *Hilbert Propositional Calculus*. We present a formulation of this system:

Rule of inference: MODUS PONENS

$$\frac{P \qquad P \to Q}{Q}$$

Axioms:

$$(P \wedge Q) \to Q$$

$$(P \wedge Q) \to P$$

$$P \to (Q \to (P \wedge Q))$$

$$P \to (P \vee Q)$$

$$Q \to (P \vee Q)$$

$$(P \to R) \to ((Q \to R) \to ((P \vee Q) \to R))$$

$$P \to (Q \to P)$$

$$(P \to Q) \to (P \to (Q \to R)) \to (P \to R)$$

$$P \to \neg\neg P$$

$$\neg\neg P \to P$$

$$(P \to Q) \to ((P \to \neg Q) \to \neg Q)$$

Note that P, Q and R are not single formulas , but meta-symbols that can stand for any propositional formula.

Let us given a Frege system $\mathcal{F}$. A $\mathcal{F}$-proof π of the formula A (we write $\pi \vdash_{\mathcal{F}} A$), is a sequence of formulas $A_1, \dots, A_m$ such that (1) each formula in the sequence is either an instance of an axiom scheme, or has been inferred by a rule from two previous formulas in the sequence, and (2) the last formula in the sequence is A. The *length* of π is the number of lines used in the sequence, i.e. m. For any propositional formula A, let $|A|$ be the number of symbols appearing in A. The size of π is defined as $\sum_{i=1}^{m} |A_i|$.

We write $A \models B$ if every truth assignment satisfying A also satisfies B, and $A \vdash B$ if adding the formula A among the axioms we can give a proof of B. A Frege proof system is *implicationally complete* if whenever $A \models B$, then $A \vdash B$, and *implicationally sound* if whenever $A \vdash B$, then $A \models B$. A Frege system will be said tree-like when we restrict ourself to proofs that can be arranged in a tree structure, i.e. when each formula in the sequence only is used once as premises of another application of a rule.

For Frege systems it is known that the tree-like case polynomially simulates the dag-like one (see [12, 51]). Very little is known about lower bounds for tautologies in Frege systems. Finding exponential lower bounds for these systems is considered a very difficult task. M.L. Bonet, S. Buss and T. Pitassi in [13] built classes of tautologies associated to some difficult combinatorial principles and showed how to prove them in Frege with at most quasipolynomial proofs. Moreover, contrary to a Cook's conjecture, S. Buss showed that the Pigeon Hole Principle tautology (used by Haken to give a lower bound in resolution) can be polynomially proved in Frege. The only known results about lower bounds in Frege were shown by S. Buss in [22] where he gave a method to obtain quadratic lower bounds for the number of symbols and linear lower bounds for the number of lines. Pudlak and Buss in [65] devised a new method, based upon interactive proofs to obtain linear lower bounds for tree-like Frege. One of the most important open problem in proof complexity is to improve these lower bounds.

Frege systems can be extended allowing extra rules, or can be restricted allowing, for example, only restricted definition of formulae. Consider the following definition: the *depth* of a formula A is the maximum number of alternations of $\{\wedge, \vee\}$ connectives in A. The *depth* of a Frege proof is the maximum depth of a formula in the proof. Between the restricted systems we mention the BOUNDED DEPTH FREGE system in which we only allow formulae of constant depth but with unbounded arity connectives. A decisive advance in proving lower bounds for Bounded depth Frege was made by Ajtai in [1], where he gave a superpolynomial lower bound for bounded depth frege proofs of the PHP_n^{n+1}. An exponential lower bound for the Pigeon Hole Principle tautology was shown by several researchers, [7, 8, 61, 56]. Particularly important is the work in [7] where they applied a technique previously used in [43] to obtain lower bounds for the size of polynomial size constant depth unbounded fan-in circuits (see section 2.2 for a more formal definition of this class of circuits).

Frege systems can be extended with some extra rules. An EXTENDED FREGE SYSTEM $e\mathcal{F}$ is a Frege system augmented with the *extension rule*. This rule is an axiom scheme of the form

$$p \leftrightarrow B$$

with the following restrictions: p is a new symbol of variable not occurring in B, in previous lines of the proof and in the last line of the proof.

Consider the following definitions. A *substitution* σ is a mapping from a finite set of propositional variables $\mathrm{Dom}(\sigma)$ to a set of well-formed formulas $\mathrm{Rng}(\sigma)$. It will be called *renaming* whenever $\mathrm{Rng}(\sigma)$ is a set of propositional variables and $\top/\bot$-*substitution* whenever $\mathrm{Rng}(\sigma) \subseteq \{\top, \bot\}$. By $A\sigma$ we denote the result of *simultaneously* replacing in A any propositional variable p_i in A with $\sigma(p_i)$.

A SUBSTITUTION FREGE SYSTEM $s\mathcal{F}$ is a Frege system with a finite number of *axioms* (instead of axiom schemes) with the further rule *substitution rule*

$$\frac{A}{A\sigma}$$

that from a formula A allows to infer the formula $A\sigma$ for any substitution σ. It will be called *Renaming Frege system* ($r\mathcal{F}$) or $\top/\bot$-*Frege system* ($\top/\bot$-$\mathcal{F}$) whenever the substitutions are respectively renamings or $\top/\bot$-substitutions.

Very little is known about lower bounds for Extended Frege and Substitution Frege. For the former system the only result is that of Buss in [22] proving linear lower bounds for the number of lines and quadratic lower bounds for the number of symbols for some constant tautologies. For the latter system Urquhart in [80] obtained a super logarithmic lower bound for the number of lines to prove some constant tautologies. In the work [15] we prove the same lower bound for another class of tautologies (see details in Chapter 3). In both cases the tree-like system polynomially simulates the general one. These results are respectively in [12, 15].

A GENTZEN PROPOSITIONAL SYSTEM is built on a sequent calculus for propositional logic. This is obtained by axiom schemes of the form $A \vdash A$, where A is a propositional formula, and a set of inference *logical* rules to manipulate introduction of the connectives and a set of *structural rules* of which it is important to mention the **cut rule**.

Gentzen systems are polynomially equivalent to Frege systems so the problem of finding exponential lower bounds for these systems is as difficult as for Frege. An important restriction of these systems is obtained if we do not allow the cut rule obtaining the **cut-free** Gentzen system.

The cut elimination theorem proves that cut-free Gentzen systems can simulate Gentzen systems but this procedure increments the symbols used of a superexponential factor. Therefore we do not have a polynomial simulation result. A. Urquhart analyzed the complexity of proofs in cut-free Gentzen system. He showed in [78] that this is not an efficient system.

2.2 Circuit complexity and Proof Complexity

In this section we introduce some of the basic models of computation for boolean functions which are related to the complexity of proof systems. The aim is not that of giving a complete introduction to circuit complexity. For a complete treatment of the subject we refer the reader to the fundamental works [19, 81].

The first model we consider is the BOOLEAN CIRCUIT MODEL. A *Boolean circuit* is a directed acyclyc graph. The nodes of in-degree 0 correspond to the inputs and are labelled by a variable or by a $O/1$ constant, the internal nodes are called gates and in this model compute a boolean function over the base $\{\wedge, \vee, \neg\}$. The in-degree and the out-degree of the nodes are referred to as *fan-in* and *fan-out*. One of the node is designated as *output* node and is considered to have fan-out 0. Observe that for different input lenghts there exist different circuits. Therefore a boolean function is computed by a family of circuits depending on the input length. In this case we speak of *non-uniform* circuits. This notion can be relaxed to an *uniformity* notion imposing that the n-th circuit of the family is easy to compute as function of n. Here we refer only to non-uniform circuits.

The fundamental parameters with respect to which we measure the complexity of a circuit computing a boolean functions are: (1) the *size*, i.e. the number of gates, and (2) the *depth*, i.e. the size of the longest path from an input to the output gate. One of the most important problems in complexity theory is to establish lower bounds for both the size and the depth of boolean circuits. This task consists of finding hard boolean functions that cannot be computed using (non-uniform) boolean circuits of a given size or depth. Boolean circuits are subdivided in classes with respect to the size and the depth. Here we consider only the definition of those classes of circuits of interest for us.

- The class NC^i, for $i \geq 0$ of the bounded fan-in boolean circuits of polynomial size and $O(\log^i n)$ depth;
- The class AC^i, for $i \geq 0$ of the unbounded fan-in boolean circuits of polynomial size and $O(\log^i n)$ depth;
- the classes $NC = AC$ of polynomial size and polylogarithmic depth circuits.
- the class $P/poly$ of boolean circuits of polynomial size.

Finding hard boolean functions that cannot be computed by circuits belonging to any of the above classes is a very difficult task. The best known result separates the class AC^0 fron the class NC^1, showing that the parity functions over n bits requires an exponential size to be computed by a circuit in AC^0 but can be computed by a circuit in the class NC^1. This was first obtained by [35]. Then [43] obtained a general technique, known as Switching Lemma, proving the same result.

An important subclass of boolean circuits is obtained when we restrict the gates only to the base $\{\wedge, \vee\}$. The class of MONOTONE CIRCUITS compute monotone boolean functions. Several results have been obtained for this subclass of circuits. For example in a very famous work A. Razborov (see [67]) obtained superpolynomial lower bounds for the size of monotone circuits computing the monotone function $CLIQUE_n^k$ deciding if a graph of n nodes has a clique of size k. Later this result was improved to an exponential factor by the work of N. Alon and R. Boppana [2]. Unfortunately Razborov himself in [68] showed that his technique, known as Approximation Method, and used in his previous work failed to extend the result to the non-monotone case.

Very recently R.Raz and P. McKenzie in [66] have shown how to separate the class monotone-NC from the class monotone-P generalizing a technique previously used by Karchmer and Wigderson in [50] to separate monotone-NC^1 from monotone-NC^2.

Pudlak in [63] introduced an extension of boolean circuits: the **real circuits**. The inputs for these circuits are boolean values, the gates compute real functions and the output must be a bit. If the real functions computed by the gates are non-decreasing then the circuit is *monotone real*. This kind of circuits was introduced in connection with problems arising in compexity of proofs. We discuss these questions in the next subsection. Here it is important to mention that for this class of circuits there has been obtained both the analogous of Razborov and Raz and McKenzie results, respectively in [63, 42] and [14].

2.2.1 Feasible Monotone Interpolation Property

Several recent results have pointed out a relationship between circuit complexity, complexity of search problems and complexity of proofs in some propositional proof systems. The main tool used is the Craig's Interpolation Theorem for propositional logic. Craig's Interpolation Theorem states that for a true implication $\phi(\vec{p},\vec{q}) \rightarrow \psi(\vec{p},\vec{r})$, where $\vec{p}\ \vec{q}\ \vec{r}$ are disjoint sets of variable, there exists a third formula $\chi(\vec{p})$ such that both the implication $\phi(\vec{p},\vec{q}) \rightarrow \chi(\vec{p})$ and $\chi(\vec{p}) \rightarrow \psi(\vec{p},\vec{r})$ are true (and thus provable). Following we shall give the main ideas of how to use it to prove the non efficiency of given propositional proof systems.

Consider $U, V \subseteq \{0,1\}^*$ two *disjoint* NP-sets, that as usual can be represented by:

$$x \in U \text{ iff } \exists y \in \{0,1\}, |y| \leq p(n) A_n(x,y)$$
$$x \in V \text{ iff } \exists z \in \{0,1\}, |z| \leq p(n) B_n(x,z)$$

where $|x| = n$ and $A_n(\vec{p},\vec{q})$ and $B_n(\vec{p},\vec{r})$ are propositional formulae of size polynomial in n. It is clear that if U and V are disjoint, then $x \notin U \cap V$ and therefore for every n we have that $\neg A_n(\vec{p},\vec{q}) \vee \neg B_n(\vec{p},\vec{r})$ is a tautology which can be rewritten as $B_n(\vec{p},\vec{r}) \rightarrow \neg A_n(\vec{p},\vec{q})$. By Craig's interpolation Theorem we know that there is an interpolant $C_n(\vec{p})$ such that both $B_n(\vec{p},\vec{r}) \rightarrow C_n(\vec{p})$ and $C_n(\vec{p}) \rightarrow \neg A_n(\vec{p},\vec{q})$ are true and thus provable. Therefore the set $L = \{x | C(x), |x| = n\}$ has the property of separating U from V. Assume we know that the sets U and V cannot be separated by any set L of a given complexity (e.g in $P/poly$). Consider a proof system F in which we can prove the following kind of result: from a proof R of $B_n(\vec{p},\vec{r}) \to \neg A_n(\vec{p},\vec{q})$ we can build an interpolant $C_n(\vec{p})$ having the size equals to the size of R. Therefore the complexity of R (i.e. its size in F) must be at least the complexity of L. The basic ideas of this approach were implicitly given firstly in the works of Krajíček and Razborov [52, 68]. Then Bonet, Pitassi and Raz in [17], employing similar ideas (that we also discuss below), showed that the Cutting Planes system with polynomially bounded coefficients is not efficient. Pudlak in [63] gave a general framework for this method obtaining as consequence the non efficiency of the Cutting Planes system (with unbounded coefficients). His main theorem can be reformulated for the Resolution systems as follows:

Theorem 2.2.1 *There is a polynomial time algorithm that, given a Resolution refutation R of*

$$\bigwedge_{i\in[n]} A_i(\vec{p},\vec{q}) \wedge \bigwedge_{i\in[n]} B_i(\vec{p},\vec{r})$$

produces a boolean circuit $C(\vec{p})$ having size equals to the size of R, computing the interpolant of $A_n(\vec{p},\vec{q})$ and $B_n(\vec{p},\vec{r})$.

The main consequence of this theorem is therefore the following: if two disjoint NP sets cannot be separated by any set in $P/poly$, then the Resolution system is not efficient. It is easy to see that if $NP \cap co \perp NP \not\subseteq P/poly$, then there are two disjoint sets that cannot be separated in $P/Poly$. But such a result is one of the most important problems in Complexity theory yet to be solved. On the other hand a monotone version of this result is known (see [2, 67]) and moreover Pudlak in [63] was able to obtain a monotone version of his theorem 2.2.1 in which the interpolant is a monotone circuit provided that the common variables $\vec{p}$ appear only positively in one formula and negatively in the other.

An example of how works Theorem 2.2.1 (in the monotone version) is the following: consider $\vec{p}$ variables encoding an undirect graph of n vertices and consider: (1) the propositional CNF formula $Colour(\vec{q},\vec{p})$ containing $\vec{p}$ only negated, expressing that the variables $\vec{q}$ encode a $k \perp 1$ coloring of the graph encoded by the $\vec{p}$ variables, and (2) the propositional

CNF formula $Clique(\vec{p}, \vec{r})$, containing the $\vec{p}$ only positively, such that the $\vec{r}$ variables encode a k-clique contained in the graph encoded by $\vec{p}$. The main result of [2] says that for $k = \Omega((n/\log n)^{2/3})$ the best monotone circuit separating the sets represented by the two formulae has size $\Omega(2^{(n/\log n)^{2/3}})$. Therefore by theorem 2.2.1 the smallest resolution refutation for $Colour(\vec{q}, \vec{p}) \wedge Clique(\vec{p}, \vec{r})$ has size $\Omega(2^{(n/\log n)^{2/3}})$. Observe that for the Cutting Planes proof system Pudlak was able to obtain the same general result given for Resolution in theorem 2.2.1 with the only difference being that the role of boolean circuits is played by real circuits.

Chapter 3

Lower and Upper Bounds for Frege Systems

The best known lower bounds for Frege ($\mathcal{F}$) and Extended Frege ($e\mathcal{F}$) are linear for the number of lines and quadratic for the number of symbols ([22]). Buss posed the open question of finding superlogarithmic lower bounds for the number of lines in $s\mathcal{F}$. In [80] Urquhart considered a class of tautologies $\top_x$ associated with a binary string x and showed that $\Omega(\frac{n}{\log n})$ lines are required to obtain $s\mathcal{F}$proofs of his tautologies,

Here we reformulate Urquhart's proof in terms of the Incompressibility Method from Kolmogorov Complexity. We re-obtain his lower bound for the class of tautologies $\top_x$ in dag-like $s\mathcal{F}$ and and we extend it to a $\Omega(n)$ lower bound in the tree-like case. The same lower bounds also hold for another class of tautologies: the permutation tautologies. We give explicit proofs for the tautologies $\top_x$ in both tree-like and dag-like $s\mathcal{F}$. This proves that the lower bounds are optimal, since we give matching upper bounds for the number of lines.

Since there is a logarithmic speed-up between tree-like and dag-like $s\mathcal{F}$ proofs of $\top_x$ and since the given bounds are optimal, we approach the question of whether tree-like $s\mathcal{F}$ simulates dag-like $s\mathcal{F}$ with a logarithmic factor of increment in the number of lines. This result is known for $\mathcal{F}$ ([12]) and we prove it for $e\mathcal{F}$. Composing the previous simulation with two more known results we show that tree-like $s\mathcal{F}$ polynomially simulates dag-like $s\mathcal{F}$:

(1) $e\mathcal{F}$ p-simulates $s\mathcal{F}$ (2) tree-like $e\mathcal{F}$ p-simulates $e\mathcal{F}$; (3)tree-like $s\mathcal{F}$ p-simulates tree-like $e\mathcal{F}$. The first simulation is showed in a work of Krajìček and Pudlàk [55], and the third is a result of [31] that we improve to $O(n \log n)$ lines instead of $O(n^2)$.

Finally we observe that $\mathcal{F}$ linearly simulates tree-like $s\mathcal{F}$ only in the number of lines. This result has as a consequence that $\mathcal{F}$ p-simulates tree-like Renaming Frege ($r\mathcal{F}$).

3.1 Further definitions

Recall the definitions of number of lines in Frege systems, and of substitutions from Chapter 2. Given two Frege systems $\mathcal{F}$ and $\mathcal{F}'$ by $\mathcal{F} \overset{f(n)}{\vdash} \mathcal{F}'$ we denote that the proof system $\mathcal{F}$ simulates the proof system $\mathcal{F}'$ with a $f(n)$ factor of increment in the number of lines.

Let $Var(A)$ be the set of variables appearing in A. Let σ and λ be two substitutions such that $\mathrm{Rng}(\lambda) \subseteq \mathrm{Dom}(\sigma)$. By $\lambda\sigma$ we denote the new substitutions θ such that $\mathrm{Dom}(\theta) = \mathrm{Dom}(\lambda)$, and if $\lambda(p_i) = A_i$, then $\theta(p_i) = A_i\sigma$. Two formulas A and B are isomorphic if there is a bijective renaming (or relettering) σ such that $A = B\sigma$. Let $\Sigma = \{A_1, \ldots, A_k\}$ be a set of formulas and $\mathrm{Var}(\Sigma) = \bigcup_{i=1}^{k} \mathrm{Var}(A_i)$. A substitution σ unifies Σ if $A_1\sigma = \ldots = A_k\sigma$ and in this case Σ is said to be *unifiable.* A substitution σ is a *most general unifier* (mgu) for Σ if and only if it unifies Σ and for any other unifier θ there is a substitution λ with $\mathrm{Dom}(\lambda) \subseteq \mathrm{Var}(\mathrm{Rng}(\sigma))$ such that $\theta = \lambda\sigma$. This means that a mgu of a set of formulas Σ is essentially unique since for any two mgu's σ_1 and σ_2 there is a bijective renaming λ such that $\sigma_1\lambda = \sigma_2$.

The following Theorem is due to Robinson (see [37] cap.8 for its proof).

Theorem 3.1.1 *There is a deterministic algorithm $\mathcal{A}$, that always halts, such that for any set Σ of formulas, if Σ is unifiable, then $\mathcal{A}$ outputs its most general unifier.*

Let A and $B \to C$ be two formulas. The rule of *condensed detachment* with premises A and $B \to C$ and conclusions D, denoted by $\mathbf{CD}(A, B \to C)$ was introduced originally in [62] and used in [44, 80]. It is defined as follows:

1. find an isomorphic formulas A' of A such that $\mathrm{Var}(A') \cap \mathrm{Var}(B \rightarrow C) = \emptyset$;

2. if $\mathcal{A}(\{A', B\}) = \sigma$, then change by a renaming σ to σ^*, in such a way that $[\mathrm{Var}(B\sigma^*) \perp \mathrm{Var}(B)] \cap \mathrm{Var}(C) = \emptyset$ and define $D = C\sigma^*$. If $\{A', B\}$ is not unifiable then $\mathbf{CD}(A, B \rightarrow C)$ is undefined.

Intuitively we can think of CD as a single rule merging together the Modus Ponens and the substitution rules. A *Condensed Detachment Frege system* $CD(\mathcal{F})$ is a Frege system whose only rule is the condensed detachment and where we use a finite number of axioms instead of axiom schemes. We suppose that the axioms used in $CD(\mathcal{F})$are indexed by an order and that in the proofs all the axioms are introduced in the first lines. It is easy to see that any proof in $CD(\mathcal{F})$can be transformed into such an equivalent one. We denote by $i = CD(j, k)$, with $j, k < i$, the fact that in a proof the formula in the line i has been obtained by the CD rule applied to formulas on line j and on line k.

The following are *Urquhart's tautologies.* Let x be a binary string, then

$$\top_x = \begin{cases} \top & \text{if } x = \epsilon \\ (\top \rightarrow \top_y) & \text{if } x = 1y \\ (\perp \vee \top_y) & \text{if } x = 0y \end{cases}$$

The *permutation tautologies* are defined as

$$\Pi_n = (p_1 \wedge (p_2 \wedge (\ldots \wedge (p_{n-1} \wedge p_n)\ldots))) \rightarrow (p_{\pi(1)} \wedge (p_{\pi(2)} \wedge (\ldots \wedge (p_{n-1} \wedge p_{\pi(n)})\ldots)))$$

where p_i are propositional variables and π is a permutation function of $[1\ldots n]$. In the rest of the Chapter, all log functions are in base two and $\mathbf{j}$ denotes the binary representation of the number j.

3.2 Lower bounds for the number of lines in $s\mathcal{F}$

In [80], the following theorem is proven:

Theorem 3.2.1 *If P is a dag-like proof in $s\mathcal{F}$, then there is a dag-like proof P' in $CD(\mathcal{F})$ such that: (1) every step in P is a substitution instance of a step in P': (2) the number of lines of P' is less than or equal to the number of lines of P.*

It is easy to see that the same theorem also holds for the tree-like case. Therefore, for classes of tautologies that are not substitution instances of shorter tautologies, we can obtain lower bounds for the number of lines in dag-like (resp. tree-like) $s\mathcal{F}$, giving lower bounds for the number of lines in dag-like (resp. tree-like) $CD(\mathcal{F})$. So, in what follows we will only work with $CD(\mathcal{F})$.
Following [80] we define the *succinct representation* for a dag-like $CD(\mathcal{F})$ proof P of m lines as a string G_P over the alphabet $\{0, 1, \#\}$. For each line i in P we define $G_P(i)$ as follows: (1) if i corresponds to the axiom j, then $G_P(i) = \mathbf{j}$; (2) if i is $CD(j, k)$, then $G_P(i) = \mathbf{j}\#\mathbf{k}$. G_P is defined as $G_P(1)\#\#G_P(2)\#\# \ldots \#\#G_P(m)$ and it is easy to see that $|G_P| = O(m \log m)$. Algorithm $\mathcal{A}$ of Theorem 3.1.1 allows us to recover uniquely the conclusion of a CD rule from the two premises, so that a proof in $CD(\mathcal{F})$ can be recovered uniquely from its succinct representation.

Theorem 3.2.2 *([80]) From G_P we can recover P uniquely (up to a relettering of the variables).*

In the tree-like case we modify the definition of succinct representation, improving its size by a logarithmic factor, but preserving the previous Theorem.

The succinct representation S_P of a tree-like $CD(\mathcal{F})$ proof P of m lines is a string over the alphabet $\{0, 1, [,], \#\}$. For each line i we define $S_P(i)$ as follows: (1) if i is the axiom j, then $S_P(i) = \mathbf{j}$; (2) if i is $CD(j, k)$, then $SP(i) = [S_P(j)\#S_P(k)]$. S_P is defined as $S_P(m)$, and it is easy to see that $|S_P| = m(3 + \log d) = O(m)$, where d is the number of axiom in $CD(\mathcal{F})$.

Theorem 3.2.3 *([65, 60]) If P is a tree-like $CD(\mathcal{F})$ proof of A, then from S_P we can recover uniquely a tree-like proof P' of A with the same number of lines of P.*

Proof. First from S_P we recover the skeleton (i.e. the tree structure) of the proof. Then starting from the leaves of the skeleton, we first recover the axioms from their index numbers, then proceeding by depth, for each internal node we recover uniquely, using the algorithm $\mathcal{A}$ of Theorem 3.1.1 the formula associated with that node. Since the number of the nodes in the skeleton is the number of lines of P, we have recovered a proof P' with the same number of lines. The last formula of P' is A since the root node of the skeleton corresponds to the last formula in P and it is the last one to be recovered. □

3.2.1 A Short Introduction to Kolmogorov Complexity

We will give an extremely brief introduction to Kolmogorov Complexity quoted from [5] and outline the idea of our lower bound proof.

The original goal of Kolmogorov Complexity was to have a quantitative measure of the complexity of a finite object. Kolmogorov and others had the following idea: the regularities of an object can be used to give a short description of it; on the other hand, if an object is highly non-regular, or random, there should be no way of describing it that is much shorter than giving the full object itself. To formalize this notion, we first encode discrete objects as strings. Second, we want to have descriptions that can be handled algorithmically, so we identify descriptions with "programs for a sufficiently powerful model of computation".

Fix a Universal Turing Machine U whose input and output alphabet is $\{0, 1\}$. The Kolmogorov complexity of a binary string x is the minimum length of a program that makes U generate x and stops. Observe that this notion seems to depend on the choice of the Universal Turing Machine. However, it can be shown that changing the machine only affects this measure of complexity by an additive constant. Strings whose Kolmogorov complexity is equal, or close to, their length are called Kolmogorov random, or *incompressible*. These are strings that cannot be compressed algorithmically. As there are at most $2^n \perp 1$ binary "programs" of length $n \perp 1$ or less, clearly there is some string of length n whose Kolmogorov complexity is at least n. For c even a small constant, this amounts to say that most strings of length n, all but a fraction of 2^{-c}, have Kolmogorov complexity $\geq n \perp c$, or are almost random (see [58] pp. 96). Many combinatorial properties have simple proofs via this prepackaged counting argument. Suppose you want to show that property $P(x)$ holds for some string x. Take a Kolmogorov random string x. Assume that $P(x)$ is false; show that this gives a way to describe x concisely. This is a contradiction. In fact, this argument usually gives proofs that $P(x)$ holds with high probability as the majority of strings are Kolmogorov random up to small constant.

In our case the property $P(x)$ will be "for the binary string x of size n, any proof of $\top_x$ in $CD(\mathcal{F})$ requires $\Omega(f(n))$ lines". We take x a random string (i.e. its shortest description is of length close to n) and suppose that $P(x)$ is false. Then we use the succinct representation of a proof of $\top_x$ to describe x succinctly, and this will give us a contradiction.

3.2.2 Lower Bounds for Urquhart's and permutation tautologies

The next Theorem shows a $\Omega(n/\log n)$ lower bound for the number of lines required in $CD(\mathcal{F})$to prove tautologies in the class $\top_x$. Our result slightly improves the the same result of [80] since we can count how many tautologies in $\top_x$are hard and also we provide a trade-off between this number and the number of lines required.

Theorem 3.2.4 *([80]) For any $c > 0$ there are at least $2^n(1 \perp 2^{-c})+1$ binary string x of size n such that any dag-like proof of $\top_x$ in $CD(\mathcal{F})$ must have more than $\lfloor \frac{(n-c)}{d \log(n-c)} \rfloor = \Omega(\frac{n}{\log n})$ lines for some constant $d > 1$.*

Proof. Fix $c > 0$. Recall that for any $CD(\mathcal{F})$ proof of n lines the size of its succinct representation is $\leq bn \log n$ for some constant b. Let $n > c + 2$ and assume that for all x of size n there exists a dag-like proof P in $CD(\mathcal{F})$ of $< \frac{(n-c)}{d \log(n-c)}$ lines with $d > 2b \log 3 > 1$. Let G_P be the succinct representation of P. So

$$\begin{aligned} |G_P| &< \tfrac{b(n-c)}{d\log(n-c)} \cdot \log(\tfrac{(n-c)}{d\log(n-c)}) \\ &= \tfrac{b(n-c)}{d\log(n-c)} \cdot [\log(n \perp c) \perp \log(d \log(n \perp c))] \end{aligned}$$

Since $d > 1$ and $n > c + 2$ we have that $\log(d \log(n \perp c)) > 0$ and therefore

$$|G_P| < \frac{b(n \perp c)}{d}$$

Fix a string x of size n whose Kolmogorov complexity is $\geq n \perp c$. Observe that, once the axioms of $CD(\mathcal{F})$ are fixed, x can be reconstructed from G_P and from a program $\mathcal{P}$ that, on input G_P: (1) recovers the proof P, (2) takes the last formula $\top_x$ and (3) rebuilds x. $|\mathcal{P}| = O(1)$ since it is independent from x. This means that the Kolmogorov complexity of x is $\leq |G_P| \log 3 + O(1) = \frac{b(n-c)}{d} \log 3 + O(1)$ (the factor $\log 3$ is due to the fact that G_P is defined over a three symbols alphabet). Since $d > 2b \log 3$ we have that the Kolmogorov complexity of x is $< \frac{(n-c)}{2} + O(1)$. Given that x is a c-incompressible binary string, its Kolmogorov Complexity is $\geq n \perp c$, therefore we have that $\frac{n-c}{2} + O(1) > n \perp c$ or $O(1) > \frac{n-c}{2}$, which is a contradiction for n sufficiently large. The result then follows since there are at least $2^n(1 \perp 2^{-c}) + 1$ binary strings of size n whose Kolmogorov Complexity is greater than $n \perp c$ (see [58] pp. 96). □

The next Theorem extends the previous result to tree-like $CD(\mathcal{F})$ giving a linear lower bound. Observe that another way to obtain this result for $s\mathcal{F}$ is using the result of Buss in [22] that $\Omega(n)$ lines are required in $\mathcal{F}$ for proof of tautologies that are not substitution instance of any shorter tautology and Theorem 3.4.4. In order to simplify the proofs we do not specify the trade-off between c and the number of lines in the next Theorems of this section. It is easy to see that this trade-off can also be obtained in a similar way as in the previous Theorem.

Theorem 3.2.5 *For any $c > 0$ there are at least $2^n(1 \perp 2^{-c}) + 1$ binary string x of size n such that any tree-like proof of $\top_x$ in $CD(\mathcal{F})$ must have $\Omega(n)$ lines.*

Proof. Fix $c > 0$ and observe that for tree-like $CD(\mathcal{F})$ proof P of m lines the size of its succinct representation S_P is $O(m)$ and apply Theorem 3.2.3 to recover the last formula of the proof. □

Permutation tautologies Π_n were introduced in [59]. It was been shown in [59] and [65] that $\Omega(n \log n)$ lines are required for tree-like $\mathcal{F}$ proofs of some of them. Here we show that $\Omega(n)$ lines are required in dag-like $s\mathcal{F}$ and $\Omega(n \log n)$ in tree-like $s\mathcal{F}$. Observe that, if we count the size of indices of variables, then the size of any Π_n is $O(n \log n)$. Therefore this last lower bound is not really an improvement of Theorem 3.2.4.

Theorem 3.2.6 *For any $c > 0$, there are at least $2^{O(n \log n)}(1 \perp 2^{-c}) + 1$ permutations π over $[1 \ldots n]$ such that any dag-like $CD(\mathcal{F})$ proof of Π_n requires $\Omega(n)$ lines.*

Proof. Fix $c > 0$. Assume that for any permutation π over $[1 \ldots n]$ there is a dag-like $CD(\mathcal{F})$ proof for the tautology associated with π with $< bn$ lines, for some constant b. The $n!$ possible tautologies obtained from the associated permutations can be encoded in $\log(n!)$ bits. By Stirling formula we have that $n! \approx \frac{n^n\sqrt{2\pi n}}{e^n}$. So $\log(n!) \approx n \log n \perp O(n) + O(\log \sqrt{n})$ that is $O(n \log n)$. This means that we can encode any permutation tautology with a string of $dn \log n$ bits for some constant d. Now consider a binary string $x \in \{0,1\}^{dn \log n}$ with Kolmogorov complexity $\geq |x| \perp c$. Let P be the proof of the permutation tautology associated with x with less of bn lines. The succinct representation G_P of P has size $O(n \log n)$. Applying the same method of Theorem 3.2.4 and noting that the length of x is $O(n \log n)$ we obtain the desired lower bound.

□

The bound for the tree-like case is obtained from the previous Theorem as in the case of Urquhart's tautologies.

Theorem 3.2.7 *For any $c > 0$, there are at least $2^{O(n \log n)}(1 \perp 2^{-c}) + 1$ permutations π over $[1 \ldots n]$ such that any tree-like $CD(\mathcal{F})$ proof of Π_n requires $\Omega(n \log n)$ lines.*

3.3 Upper bound for Urquhart's tautologies

We know that some tautologies $\top_x$ associated with binary strings of size n require $\Omega(\frac{n}{\log n})$ lines $s\mathcal{F}$. Here we show how to obtain a dag-like $s\mathcal{F}$ proof in $O(\frac{n}{\log n})$ lines. Moreover we show that also the tree-like lower bound for T_x's is optimal since we give a $O(n)$ line tree-like $s\mathcal{F}$ proof for them.

Consider the following formulas associated with a binary string x:

$$\top^p_x = \begin{cases} p & \text{if } x = \epsilon \\ (\top \to \top^p_y) & \text{if } x = 1y \\ (\bot \vee \top^p_y) & \text{if } x = 0y \end{cases}$$

Lemma 3.3.1 *All tautologies $p \to \top^p_x$ for any $x \in \{0,1\}^n$ with $n > 0$ can be obtained in a $s\mathcal{F}$ proof of $O(2^n)$ lines.*

Proof. First observe that $p \to \top^p_1 = p \to (\top \to p)$ and $p \to \top^p_0 = p \to (\bot \vee p)$ can be obtained in a constant number of steps. Now suppose we have in the proof all the tautologies $p \to \top^p_y$ for $|y| = n \perp 1$. To obtain $p \to \top^p_x$ for $x = y1$ (resp. $x = y0$) and $|x| = n$ we do following steps:

$p \to \top^p_y$	by ind. hyp.
$(\top \to p) \to \top^p_{y1}$	subst. p with $(\top \to p)$
$p \to \top^p_{y1}$	cut with $p \to (\top \to p)$ (resp. $p \to (\bot \vee p)$)

Each one of the $p \to \top^p_x$'s can be obtained in a constant number of lines from one of the $p \to \top^p_y$'s and one between the first tautologies, where $x = y1$. So the total numbers of lines to obtain all the $\top^p_x$ is $c\sum_{i=1}^n 2^i = O(2^n)$. $\Box$

Theorem 3.3.1 *Given $x \in \{0,1\}^n$ there is a $s\mathcal{F}$ proof of $\top_x$ in $O(\frac{n}{\log n})$ lines.*

Proof. The proof is divided in two parts. First, by previous Lemma, we obtain a proof of all tautologies $p \to \top^p_y$ for all $|y| = \log(\frac{n}{\log n})$. This part requires $O(\frac{n}{\log n})$ lines.
Second, to obtain $p \to \top^p_x$ divide x in $\frac{n}{\log(\frac{n}{\log n})} = \frac{n}{\log n - \log\log n}$ substrings $x_1, \ldots, x_k$ (with $k = \frac{n}{\log n - \log\log n}$) of size $\log(\frac{n}{\log n})$. In the first part we have already proved the tautologies $p \to \top^p_{x_i}$ for all i. We put them together to form $p \to \top^p_x$ starting from the innermost one in the following way: consider the sequence $p \to \top^p_{x_1}, \ldots, p \to \top^p_{x_k}$. Let $\overline{\top}^p_{x_{k-1}}$ be the formula

obtained substituting p by $\top^p_{x_k}$ in $\top^p_{x_{k-1}}$. In a constant number of lines we obtain the formula $p \rightarrow \overline{\top}^p_{x_{k-1}}$ the following way:

$$\begin{array}{ll} p \rightarrow \top^p_{x_{k-1}} & \\ \top^p_{x_k} \rightarrow \overline{\top}^p_{x_{k-1}} & \text{subst. } p \text{ with } \top^p_{x_k} \\ p \rightarrow \overline{\top}^p_{x_{k-1}} & \text{cut with } p \rightarrow \top^p_{x_k} \end{array}$$

Now consider the sequence $p \rightarrow \top^p_{x_1}, \ldots, p \rightarrow \top^p_{x_{k-2}}, p \rightarrow \overline{\top}^p_{x_{k-1}}$ and iterate the previous steps. After $k = \frac{n}{\log n - \log\log n}$ iterations we obtain $p \rightarrow \top^p_x$ and then by replacing p by $\top$ and one instance of Modus Ponens we obtain $\top_x$. This second part requires $O(\frac{n}{\log n - \log\log n}) = O(\frac{n}{\log n})$ lines. The total number of lines is $O(\frac{n}{\log n})$. $\square$

We give a tree-like $\mathcal{F}$ proof of $O(n)$ lines for any of the 2^n formulas $\top_x$ associated with binary strings x of size n.

Theorem 3.3.2 *For any $x \in \{0,1\}^n$, there is a tree-like proof of $\top_x$ of $O(n)$ lines.*

Proof. By induction on x. The base case is trivial. Let x be $1y$. Then we have a proof of $\top_y$ in $c(n \perp 1)$ lines; introduce by an axiom, the line $\top_y \rightarrow (\top \rightarrow \top_y)$ and cut with $\top_y$. If x is $0y$, then use, in the same way, the axiom $\top_y \rightarrow (\top \vee \top_y)$. The proof of $\top_x$ is $c(n \perp 1) + 2$, and for $c \geq 2$ is less than or equal to cn. $\square$

3.4 Simulations for Frege systems with substitutions

Optimal lower and upper bounds for $s\mathcal{F}$ proofs of $\top_x$ seem to suggest that tree-like $s\mathcal{F} \vdash^{O(n \log n)}$ dag-like $s\mathcal{F}$. This is also supported by the fact that the same result holds for $\mathcal{F}$ [12] and for $e\mathcal{F}$ (see 3.4.1). But the technique used in [12], which can also be applied to $e\mathcal{F}$, does not work for $s\mathcal{F}$. Moreover, note that until now it was not known whether tree-like $s\mathcal{F}$ p-simulates dag-like $s\mathcal{F}$. In this section we solve this problem.

3.4.1 Tree-like $s\mathcal{F}$ simulation of $s\mathcal{F}$

We prove the following Theorem.

Theorem 3.4.1 *tree-like $s\mathcal{F}$ p-simulates dag-like $s\mathcal{F}$.*

Proof. We compose the following three simulations: (1) $e\mathcal{F}$ p-simulates $s\mathcal{F}$; (2) tree-like $e\mathcal{F}$ p-simulates $e\mathcal{F}$; (3) tree-like $s\mathcal{F}p$-simulates tree-like $e\mathcal{F}$. The first one is by [55]. The second one is obtained below in Theorem 3.4.2. Finally the third one is obtained in [31], and we improve it in Theorem 3.4.3. □

Consider the following formulas introduced in [11]:

Definition 3.4.1 *Let $A_1, \dots, A_n$ be formulas with n a power of 2. The Balanced conjunction $\bigwedge_{i=1}^{n} A_i$ of $A_1, \dots, A_n$ is defined inductively by*

- *if $n = 1$, then $\bigwedge_{i=1}^{n} A_i$ is A_1;*
- *otherwise, $(\bigwedge_{i=1}^{n/2} A_i) \wedge (\bigwedge_{i=1}^{n/2} A_{n/2+i})$.*

Definition 3.4.2 *Let $A_1, \dots, A_n$ be formulas with $n = 2^m + s$ with $0 < s \leq 2^m$. The Psuedobalanced conjunction $\bigwedge_{i=1}^{n} A_i$ of $A_1, \dots, A_n$ is defined inductively by*

- *if $n = 1$,then $\bigwedge_{i=1}^{n} A_i$ is A_1;*
- *otherwise, $(\bigwedge_{i=1}^{2^m} A_i) \wedge (\bigwedge_{i=1}^{s} A_{(2^s+i)})$ where the first conjunct is balanced and the second pseudobalenced.*

From this point on all conjuncts $\bigwedge$ are intended to be pseudobalanced. The following Lemma is from [12]:

Lemma 3.4.1 ([12]) *The formula $(\bigwedge_{i=1}^{k-1} A_i) \wedge A_k \rightarrow (\bigwedge_{i=1}^{k} A_i)$, where the conjunctions are associated in a pseudobalanced way, has a tree-like $\mathcal{F}$proof of $O(\log k)$ lines.*

Tree-like $e\mathcal{F}$ simulation of $e\mathcal{F}$.

This proof is similar to the analogous theorem for $\mathcal{F}$ proved in [12]. We only sketch it.

Theorem 3.4.2 *Tree-like $e\mathcal{F}$ p-simulates dag-like $e\mathcal{F}$. Moreover the following result holds: tree-like $e\mathcal{F}$ $\mid\!\underline{\;O(n\log n)\;}$ dag-like $e\mathcal{F}$.*

Proof. Let $P = A_1, \dots A_n$ be a proof in $e\mathcal{F}$. Let $B_i = \bigwedge_{j=i}^{n} A_j$ for $i = 1, \dots , n$ and $B_0 = \top$. The technique is that of obtaining separate tree-like proofs of $B_i \rightarrow B_{i+1}$ for all $i = 1 \dots n \perp 1$ in $O(\log i)$ lines depending on how A_{i+1} is inferred in P. Then prove in a tree-like way $B_n \rightarrow A_n$ and finally get A_n performing cuts between the previous proof. We treat only the case in which A_{i+1} is inferred by the extension rule. Assume that A_{i+1} is $p_k \leftrightarrow C_k$. Then obtain $B_i \rightarrow B_i$ in a constant number of steps, introduce $p_k \leftrightarrow C_k$ (this is correct since p_k never occurs in $B_0, \dots , B_i$) and in a constant number of steps obtain $B_i \rightarrow (B_i \wedge (p_k \leftrightarrow C_k))$. By previous Lemma we have $(B_i \wedge (p_k \leftrightarrow C_k)) \rightarrow B_{i+1}$ in $O(\log i)$ lines and finally $B_i \rightarrow B_{i+1}$ cutting with the formula previously obtained. $\Box$.

Tree-like $s\mathcal{F}$ simulation of tree-like $e\mathcal{F}$.

Cook and Reckhow show in [31] that $s\mathcal{F}$ p-simulates $e\mathcal{F}$. Here we show that this simulation is preserved also in the tree-like case and improve from $O(n^2)$ to $O(n \log n)$ the number of lines used.

Theorem 3.4.3 *Tree-like $s\mathcal{F}$ p-simulates tree-like $e\mathcal{F}$. Moreover the following result holds: tree-like $s\mathcal{F}$ $\vdash^{\underline{O(n \log n)}}$ tree-like $e\mathcal{F}$.*

Proof. Let P be the $e\mathcal{F}$ tree-like proof $A_1, \dots , A_n$, and suppose that k many of the A_i's are formulas of the form $p_j \leftrightarrow B_j$ introduced by the extension rule. Consider the formula B defined by $\bigwedge_{j=1}^{k}(p_j \leftrightarrow B_j)$ following the order of introduction of the extension rules. First we give a tree-like $\mathcal{F}$proof of $B \rightarrow A$. This is obtained by showing that for all $i = 1, \dots , n$ there is a proof of $B \rightarrow A_i$.

case 1: A_i is an axiom, then there is a tree-like $\mathcal{F}$proof of $B \rightarrow A_i$ in a constant number of lines;

case 2: A_i is obtained by *Modus Ponens* from A_j and A_k with $j, k < i$. We can therefore assume there are tree-like $\mathcal{F}$proofs of $B \rightarrow A_j$ and $B \rightarrow (A_j \rightarrow A_i)$. A tree-like proof of $B \rightarrow A_i$ can be easy obtained in a constant number of lines.

case 3: It is easy to see that $B \rightarrow (p_l \leftrightarrow B_l)$ in $O(\log k)$ lines.

This first part requires $O(n \log k)$ lines. Now we obtain A by the following steps:

1. from Lemma 3.4.1 obtain $\bigwedge_{j=1}^{k-1}(p_j \leftrightarrow B_j) \wedge (p_k \leftrightarrow B_k) \rightarrow B$ in $O(\log k)$ lines;
2. Cut this formula with $B \rightarrow A$ and obtain $\bigwedge_{j=1}^{k-1}(p_j \leftrightarrow B_j) \wedge (p_k \leftrightarrow B_k) \rightarrow A$;
3. substitute B_k for p_k in the last formula, derive the axiom $B_k \rightarrow B_k$ and then obtain $\bigwedge_{j=1}^{k-1}(p_j \leftrightarrow B_j) \rightarrow A$;
4. iterate this process substituting the p_i in the reverse order respect to their introduction in P.

This second part requires $O(k \log k)$ lines. Since $k \leq n$, the total number of lines is $O(n \log n)$.
$\square$

3.4.2 Tree-like $\mathcal{F}$ simulations of tree-like $r\mathcal{F}$ and tree-like $\top/\bot$-$\mathcal{F}$

In this subsection we first observe that tree-like $\mathcal{F} \mathrel{|\!\overset{O(n)}{\overline{}}}$ tree-like $s\mathcal{F}$ and then we discuss the differences between $s\mathcal{F}$, $r\mathcal{F}$ and $\top/\bot$-$\mathcal{F}$.

The technique of pushing substitution lines up above Modus Ponens lines (stated e.g. in Lemma 1.11 of [44]) gives a simple of the following Theorem

Theorem 3.4.4 *Tree-like* $\mathcal{F} \mathrel{|\!\overset{O(n)}{\overline{}}}$ *tree-like* $s\mathcal{F}$.

On a m size, k lines $s\mathcal{F}$ proof, the previous theorem provide a $\mathcal{F}$ proof with axiom formulas of size $O(m^{k+1})$. So we cannot conclude that $\mathcal{F}$ p-simulates tree-like $s\mathcal{F}$. However, an immediate corollary of this last Theorem is that any lower bound for the number of lines for tree-like $\mathcal{F}$ must also hold for tree-like $s\mathcal{F}$.

Moreover, note that if we restrict the substitutions to renamings or to $\top/\bot$-substitutions, then every application of the substitution rule in the proof does not add any new symbol. Therefore we have:

Theorem 3.4.5 *([47]) $\mathcal{F}$ p-simulates tree-like $r\mathcal{F}$ and tree-like $\top/\bot$-$\mathcal{F}$.*

This result, though corollary of an easy observation, is interesting. It is strongly believed that $\mathcal{F}$ is exponentially separated from $s\mathcal{F}$ and $e\mathcal{F}$. If this was true, then there should be some formula hard to prove in tree-like $r\mathcal{F}$ but easy to prove in dag-like $r\mathcal{F}$. Moreover, we have proved that in the case of a full substitution this is not true, since the tree-like case is as powerful as the dag-like one; and this result also holds for $\mathcal{F}$ and $e\mathcal{F}$.

Chapter 4

Exponential Separations for Resolution and Cutting Planes

A commonly used strategy for finding proofs is to reduce the search space by defining restricted versions of resolution that are still complete. One possibility is to restrict the system to proofs that are tree-like. In this Chapter we prove (Corollary 4.2.1) an exponential separation between tree-like resolution and resolution, hence showing that finding tree-like resolution proofs cannot be an efficient strategy for finding resolution proofs. Until now only superpolynomial separations were known [79, 30].

Many strategies for finding resolution proofs are described in the literature, see e.g. Schöning's textbook [74], but very little theoretical work has been done until now. Here we consider three of the most commonly used restrictions of resolution: N-resolution, Regular and Davis-Putnam resolution. We show an exponential separation between tree-like Resolution and each one of the above restrictions (Corollary 4.2.1 for N-resolution and Corollary 4.2.2 for Regular and Davis-Putnam). Goerdt [38, 39, 40] gave several superpolynomial separations between resolution and some restricted versions of it. In particular, he gave a separation between Davis-Putnam resolution and unrestricted resolution. We improve this result by giving an exponential separation between Davis-Putnam and N-resolution (Corollary 4.3.1), showing that using the Davis-Putnam restriction is not, in general, a good strategy for finding resolution proofs.

The task of finding hard-to-prove tautologies in the Cutting Planes proof system is solved. Impagliazzo et al. [45] were the first to show such a result in the restricted case of CP proofs whose underlying graph is a tree. Bonet et al. [17] proved a lower bound for a CP system that did not have the tree-like restriction, but put a restriction in the size of the coefficients. Finally Pudlák [63] and Cook and Haken [42] gave general circuit complexity results from which a exponential lower bounds for CP follow.

Nothing is known about automatizability of CP proofs. Since there is an exponential speed-up of CP over Resolution, it would be nice to find an efficient algorithm for finding CP proofs. A question to ask is if trying to find tree-like CP proofs would be an efficient strategy for finding Cutting Planes proofs.

Johannsen [48] gave a superpolynomial separation, with a lower bound of the form $\Omega(n^{\log n})$, between tree-like CP and dag-like CP (this was previously known for a restricted form of CP from [17]). In this Chapter we improve that separation to exponential (Corollary 4.2.1). This means again that trying to find tree-like proofs is not a good strategy.

This exponential separation is a consequence of extending the lower bounds of Raz and McKenzie [66] to the case of monotone real circuits. As in [66], we prove an $\Omega(n^{\epsilon})$ lower bound on the depth of monotone real circuits computing a certain monotone function $\textsc{Gen}_n$ in P. This also implies an $\Omega(2^{n^{\epsilon}})$ lower bound on the size of monotone real formulas computing $\textsc{Gen}_n$. This latter result allows us to obtain an exponential lower bound for the size of tree-like CP proofs for a formula associated to $\textsc{Gen}_n$, using the interpolation technique of [52, 63] (Theorem 4.2.2).

We start by showing the result about monotone real circuits.

4.1 Monotone Real Circuits

A *monotone real circuit* is a circuit of fan-in 2 computing with real numbers where every gate computes a nondecreasing real function. This class of circuits was introduced by Pudlák

[63]. We require that monotone real circuits output 0 or 1 on every input of zeroes and ones only, so that they are a generalization of monotone boolean circuits. The depth and size of a monotone real circuit are defined as usual, and we call it a *formula* if every gate has fan-out at most 1.

Lower bounds on the size of monotone real circuits were given by Pudlák [63], Cook and Haken [42] and Fu [34]. Rosenbloom [73] shows that they are strictly more powerful than monotone boolean circuits, since every slice function can be computed by a linear size, logarithmic depth monotone real circuit, whereas most slice functions require exponential size general boolean circuits. On the other hand, Jukna [49] gives a general lower bound criterion for monotone real circuits, and uses it to show that certain functions in $P/poly$ require exponential size monotone real circuits. Hence the computing power of monotone real circuits and general boolean circuits is incomparable.

For a monotone boolean function f, we denote by $d_{\mathbb{R}}(f)$ the minimal depth of a monotone real circuit computing f, and by $s_{\mathbb{R}}(f)$ the minimal size of a monotone real formula computing f.

The method of proving lower bounds on the depth of monotone boolean circuits using communication complexity was used by Karchmer and Wigderson [50] to give an $\Omega(\log^2 n)$ lower bound on the monotone depth of st-connectivity. Using the notion of real communication complexity introduced by Krajíček [53], Johannsen [48] proved the same lower bound for monotone real circuits.

The monotone function $\textsc{Gen}_n$ of n^3 inputs $t_{a,b,c}$, $1 \leq a, b, c \leq n$ is defined as follows: For $c \leq n$, we define the relation $\vdash c$ (*c is generated*) recursively by

$$\vdash c \quad \text{iff} \quad c = 1 \text{ or there are } a, b \leq n \text{ with } \vdash a \text{ , } \vdash b \text{ and } t_{a,b,c} = 1 \ .$$

Finally $\textsc{Gen}_n(\vec{t}) = 1$ iff $\vdash n$. From now on we will write $a, b \vdash c$ for $t_{a,b,c} = 1$.

Recently, Raz and McKenzie [66] proved a lower bound of the form $\Omega(n^{\epsilon})$ for some $\epsilon > 0$ on the depth of monotone boolean circuits computing $\textsc{Gen}_n$. We will show that their method applies to monotone real circuits:

Theorem 4.1.1 *For some $\epsilon > 0$ and sufficiently large n*

$$d_{\mathbb{R}}(\mathrm{GEN}_n) \geq \Omega(n^\epsilon) \quad \textit{and} \quad s_{\mathbb{R}}(\mathrm{GEN}_n) \geq 2^{\Omega(n^\epsilon)} \ .$$

4.1.1 Real Communication Complexity

Let $R \subseteq X \times Y \times Z$ be a multifunction, i.e. for every pair $(x, y) \in X \times Y$, there is a $z \in Z$ with $(x, y, z) \in R$. We view such a multifunction as a search problem, i.e., given input $(x, y) \in X \times Y$, the goal is to find a $z \in Z$ such that $(x, y, z) \in R$.

A real communication protocol over $X \times Y \times Z$ is executed by two players I and II who exchange information by simultaneously playing real numbers and then comparing them according to the natural order of $\mathbb{R}$. This generalizes ordinary deterministic communication protocols in the following way: in order to communicate a bit, the sender plays this bit, while the receiver plays a constant between 0 and 1, so that he can determine the value of the bit from the outcome of the comparison.

Formally, such a protocol P is specified by a binary tree, where each internal node v is labeled by two functions $f_v^I : X \to \mathbb{R}$, giving player I's move, and $f_v^{II} : Y \to \mathbb{R}$, giving player II's move, and each leaf is labeled by an element $z \in Z$. On input $(x, y) \in X \times Y$, the players construct a path through the tree according to the following rule:

> At node v, if $f_v^I(x) > f_v^{II}(y)$, then the next node is the left son of v, otherwise the right son of v.

The value $P(x, y)$ computed by P on input (x, y) is the label of the leaf reached by this path.

A real communication protocol P solves a search problem $R \subseteq X \times Y \times Z$ if for every $(x, y) \in X \times Y$, $(x, y, P(x, y)) \in R$ holds. The *real communication complexity* $CC_{\mathbb{R}}(R)$ of a search problem R is the minimal depth of a real communication protocol that solves R.

For a natural number n, let $[n]$ denote the set $\{1, \ldots, n\}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function, let $X := f^{-1}(1)$ and $Y := f^{-1}(0)$, and let the multifunction $R_f \subseteq X \times Y \times [n]$ be defined by

$$(x, y, i) \in R_f \quad \text{iff} \quad x_i = 1 \text{ and } y_i = 0$$

The Karchmer-Wigderson game for f is defined as follows: Player I receives an input $x \in X$ and Player II an input $y \in Y$. They have to agree on a position $i \in [n]$ such that $(x, y, i) \in R_f$. Sometimes we will say that R_f is the Karchmer-Wigderson game for the function f. There is a relation between the real communication complexity of R_f and the depth of a monotone real circuit or the size of a monotone real formula computing f, similar to the boolean case:

Lemma 4.1.1 (Krajíček [53]) *Let f be a monotone boolean function. Then*

$$CC_{\mathbb{R}}(R_f) \leq d_{\mathbb{R}}(f) \text{ and } CC_{\mathbb{R}}(R_f) \leq \log_{3/2} s_{\mathbb{R}}(f) .$$

For a proof see [53] or [48]. Hence to establish Theorem 4.1.1, it suffices to prove:

Theorem 4.1.2 *For some $\epsilon > 0$ and sufficiently large n*

$$CC_{\mathbb{R}}(R_{\mathrm{GEN}_n}) \geq \Omega(n^{\epsilon}).$$

4.1.2 DART games and structured protocols

Raz and McKenzie [66] introduced a special kind of communication games, called DART games, and a special class of communication protocols, the *structured protocols*, for solving them.

For $m, k \in \mathbb{N}$, the set of communication games $\mathrm{DART}(m, k)$ is defined as follows:

- $X = [m]^k$. I.e., the inputs for Player I are k-tuples of elements $x_i \in [m]$.
- $Y = (\{0,1\}^m)^k$. I.e., the inputs for Player II are k-tuples of binary colorings y_i of $[m]$.
- For all $i = 1, \ldots, k$ let $e_i = y_i(x_i)$. The relation $R \subseteq X \times Y \times Z$ defining the game only depends on $e_1, \ldots, e_k$ and z, i.e., we can describe $R(x, y, z)$ as $R((e_1, \ldots, e_k), z)$.
- $R((e_1, \ldots, e_k), z)$ is a DNF-Search-Problem, i.e., there exists a DNF-tautology F_R defined over the variables $e_1, \ldots, e_k$ such that Z is the set of terms of F_R, and $R((e_1, \ldots, e_k), z)$ holds if and only if the term z is satisfied by the assignment $(e_1, \ldots, e_k)$.

A *structured protocol* for a DART game is a communication protocol for solving the search problem R, where player I gets input $x \in X$, player II gets input $y \in Y$, and in each round, player I reveals the value x_i for some i, and II replies with $y_i(x_i)$. The structured communication complexity of $R \in \mathrm{DART}(m,k)$, denoted by $SC(R)$, is the minimal number of rounds in a structured protocol solving R.

The main theorem of [66] showed that for suitable m and k, the deterministic communication complexity of a DART game cannot be much smaller than that of a structured protocol. We shall show the same for its real communication complexity. Obviously, a structured protocol solving R in r rounds can be simulated by a real communication protocol solving R in $r \cdot (\lceil \log m \rceil + 1)$ rounds. Conversely, we will prove that the following holds:

Theorem 4.1.3 *For every relation $R \in \mathrm{DART}(m,k)$, where $m \geq k^{14}$,*

$$CC_{\mathbb{R}}(R) \geq SC(R) \cdot \Omega(\log m) .$$

In particular, this implies that for DART games, real communication protocols are no more powerful than deterministic communication protocols.

Corollary 4.1.1 *For $R \in \mathrm{DART}(m,k)$ with $m \geq k^{14}$,*

$$CC_{\mathbb{R}}(R) = \Theta(CC(R)) .$$

This follows from the chain of inequalities

$$CC(R) \geq CC_{\mathbb{R}}(R) \geq SC(R) \cdot \Omega(\log m) \geq \Omega(CC(R)) .$$

4.1.3 A DART game related to $\textsc{Gen}_n$

The communication game $\textsc{PyrGen}(m,d)$ is defined as follows:

Let $Pyr_d := \{(i,j)\,;\, 1 \leq j \leq i \leq d\}$. We regard the indices as elements of Pyr_d, so that the inputs for the two players I and II are respectively sequences of elements $x_{i,j} \in [m]$ and $y_{i,j} \in \{0,1\}^m$ with $(i,j) \in Pyr_d$, and we picture these as laid out in a pyramidal form with $(1,1)$ at the top and (d,j), $1 \leq j \leq d$ and the bottom. The goal of the game is to find either an element colored 0 at the top of the pyramid, or an element colored 1 at the bottom of the pyramid, or an element colored 1 with the two elements below it colored 0, i.e. to find indices (i, j) such that one of the following holds:

1. $i = j = 1$ and $y_{1,1}(x_{1,1}) = 0$, or
2. $y_{i,j}(x_{i,j}) = 1$ and $y_{i+1,j}(x_{i+1,j}) = 0$ and $y_{i+1,j+1}(x_{i+1,j+1}) = 0$, or
3. $i = d$ and $y_{d,j}(x_{d,j}) = 1$.

Obviously, $\text{PyrGen}(m, d)$ is a game in $\text{DART}(m, \binom{d+1}{2})$. A lower bound on the structured communication complexity of $\text{PyrGen}(m, d)$ was proved in [66]:

Lemma 4.1.2 *$SC(\text{PyrGen}(m, d)) \geq d$.*

Hence by Theorem 4.1.3, we get $CC_{\mathbb{R}}(\text{PyrGen}(m, d)) \geq \Omega(d \log m)$ for $m \geq d^{28}$.

The following reduction shows that the real communication complexity of the game $\text{PyrGen}(m, d)$ is bounded by the real communication complexity of the Karchmer-Wigderson game for Gen_n for a suitable n. The proof of the next theorem follows the ideas of [66].

Lemma 4.1.3 *For $n := m \cdot \binom{d+1}{2} + 2$,*

$$CC_{\mathbb{R}}(\text{PyrGen}(m, d)) \leq CC_{\mathbb{R}}(R_{\text{Gen}_n}).$$

Proof: We prove that any protocol P solving the Karchmer-Wigderson game for Gen_n can be used to solve the $\text{PyrGen}(m, d)$ game. From their respective inputs for the $\text{PyrGen}(m, d)$ game, Player I and II compute respectively a minterm and a maxterm for Gen_n and then apply the protocol P.

We interpret the elements between 2 and $n \perp 1$ as triples (i, j, k), where $(i, j) \in Pyr_d$ and $k \in [m]$.

Now player I computes from his input $x : Pyr_d \to [m]$ an input $\vec{t_x}$ to $\textsc{Gen}_n$ with $\textsc{Gen}_n(\vec{t_x}) = 1$ by setting the following:

$$\begin{array}{ll} 1,1 \vdash a_{d,j} & \text{for } 1 \leq j \leq d \\ a_{1,1}, a_{1,1} \vdash n & \\ a_{i+1,j}, a_{i+1,j+1} \vdash a_{i,j} & \text{for } (i,j) \in Pyr_{d-1} \end{array}$$

where $a_{i,j} := (i, j, x_{i,j})$. This completely determines $\vec{t_x}$.

Likewise Player II computes from his input $y : Pyr_d \to (2^{[m]})$ a coloring col of the elements from $[n]$ by setting $col(1) = 0$, $col(n) = 1$ and $col((i,j,k)) = y_{i,j}(k)$. From this, he computes an input $\vec{t_y}$ by setting $a, b \vdash c$ iff it is not the case that $col(c) = 1$ and $col(a) = col(b) = 0$. Obviously $\textsc{Gen}_n(\vec{t_y}) = 0$.

Playing the Karchmer-Wigderson game for $\textsc{Gen}_n$ now yields a triple (a, b, c) such that $a, b \vdash c$ in $\vec{t_x}$ and $a, b \not\vdash c$ in $\vec{t_y}$. By definition of $\vec{t_y}$, this means that $col(a) = col(b) = 0$ and $col(c) = 1$, and by definition of $\vec{t_x}$ one of the following cases must hold:

- $a = b = 1$ and $c = a_{d,j}$ for some $j \leq d$. By definition of col, $y_{d,j}(x_{d,j}) = 1$.
- $c = n$ and $a = b = a_{1,1}$. In this case, $y_{1,1}(x_{1,1}) = 0$.
- $a = a_{i+1,j}$, $b = a_{i+1,j+1}$ and $c = a_{i,j}$. Then we have $y_{i,j}(x_{i,j}) = 1$, and $y_{i+1,j}(x_{i+1,j}) = y_{i+1,j+1}(x_{i+1,j+1}) = 0$.

In either case, the players have solved $\textsc{PyrGen}(m, d)$ without any additional communication.

$\square$

Now the lower bound on $CC_{\mathbb{R}}(\textsc{PyrGen}(m, d))$ obtained from Lemma 4.1.2 and Theorem 4.1.3, together with Lemma 4.1.3 immediately imply Theorem 4.1.2 with $\epsilon = \frac{1}{30}$ by taking $m = d^{28}$.

Let $\vec{t}$ be an input to $\textsc{Gen}_n$. We say that n is generated in a depth-d pyramidal fashion by $\vec{t}$ if there is a mapping $m : Pyr_d \to [n]$ such that the following hold (recall that $a, b \vdash c$

means $t_{a,b,c} = 1$):

$$
\begin{array}{ll}
1, 1 \vdash m(d, j) & \text{for every } j \le d \\
m(i+1, j), m(i+1, j+1) \vdash m(i, j) & \text{for every } (i, j) \in Pyr_{d-1} \\
m(1, 1), m(1, 1) \vdash n &
\end{array}
$$

As the reduction in the proof of Lemma 4.1.3 produces only inputs from $\text{GEN}_n^{-1}(1)$ which have the additional property that n is generated in a depth-d pyramidal fashion, we can state the following strengthening of Theorem 4.1.1:

Corollary 4.1.2 *Let n, d and ϵ be as above. Every monotone real formula that outputs 1 on every input to GEN_n for which n is generated in a depth-d pyramidal fashion, and outputs 0 on all inputs where GEN_n is 0, has to be of size $\Omega(2^{n^\epsilon})$.*

The other consequences drawn from Theorem 4.1.3 and Lemma 4.1.2 in [66] apply to monotone real circuits as well, e.g. we just state without proof the following result:

Theorem 4.1.4 *There are constants $\epsilon, c > 0$ such that for every function $d(n) \le n^\epsilon$, there is a family of monotone functions $f_n : \{0,1\}^n \to \{0,1\}$ that can be computed by monotone boolean circuits of size $n^{O(1)}$ and depth $d(n)$, but cannot be computed by monotone real circuits of depth less than $c \cdot d(n)$.*

The method also gives a simpler proof of the lower bounds in [48], in the same way as [66] simplifies the lower bound of [50].

4.1.4 Proof of Theorem 4.1.3

To prove Thm. 4.1.3, we first need some combinatorial notions and results from [66]. Let $A \subseteq [m]^k$ and $1 \le j \le k$. For $x \in [m]^{k-1}$, let $\deg_j(x, A)$ be the number of $\xi \in [m]$ such that

$(x_1, \ldots, x_{j-1}, \xi, x_j, \ldots, x_{k-1}) \in A$. Then we define

$$\begin{aligned} A[j] &:= \left\{ x \in [m]^{k-1} \,;\, \deg_j(x, A) > 0 \right\} \\ AVDEG_j(A) &:= \frac{|A|}{|A[j]|} \\ MINDEG_j(A) &:= \min_{x \in A[j]} \deg_j(x, A) \\ Thickness(A) &:= \min_{1 \le j \le k} MINDEG_j(A)\,. \end{aligned}$$

The following lemmas about these notions were proved in [66]:

Lemma 4.1.4 *For every $A' \subseteq A$ and $1 \le j \le k$,*

$$AVDEG_j(A') \ge \frac{|A'|}{|A|} AVDEG_j(A) \tag{4.1}$$

$$Thickness(A[j]) \ge Thickness(A) \tag{4.2}$$

Lemma 4.1.5 *If for every $1 \le j \le k$, $AVDEG_j(A) \ge \delta m$ for some $0 < \delta < 1$, then for every $\alpha > 0$ there is $A' \subseteq A$ with $|A'| \ge (1 \perp \alpha)|A|$ and*

$$Thickness(A') \ge \frac{(1 \perp \alpha)\delta m}{k\big(1 + \alpha^{-1}\ln(\delta^{-1})\big)}\,.$$

In particular, setting $\alpha = \frac{1}{2}$ and $\delta = 4m^{-\frac{1}{14}}$, we get

Corollary 4.1.3 *If $m \ge k^{14}$ and for every $1 \le j \le k$, $AVDEG_j(A) \ge 4m^{\frac{13}{14}}$, then there is $A' \subseteq A$ with $|A'| \ge \frac{1}{2}|A|$ and $Thickness(A) \ge m^{\frac{11}{14}}$.*

For a relation $R \in \mathrm{DART}(m, k)$, $A \subseteq X$ and $B \subseteq Y$, let $CC_{\mathbb{R}}(R, A, B)$ be the real communication complexity of R restricted to $A \times B$.

Fix a large $m \in \mathbb{N}$. A triple (R, A, B) is called an (α, β, ℓ)-game if $R \in \mathrm{DART}(m, k)$ for some $k \le m^{\frac{1}{14}}$ with $SC(R) \ge \ell$, $A \subseteq X$ with $|A| \ge 2^{-\alpha}|X|$ and $Thickness(A) \ge m^{\frac{11}{14}}$, and $B \subseteq Y$ with $|B| \ge 2^{-\beta}|Y|$.

Lemma 4.1.6 *For every $\alpha, \ell \ge 0$ and $0 \le \beta \le m^{\frac{1}{7}}$ and every (α, β, ℓ)-game (R, A, B),*

1. *if for every* $1 \leq j \leq k$, $AVDEG_j(A) \geq 8m^{\frac{13}{14}}$, *then there is an* $(\alpha+2, \beta+1, \ell)$*-game* (R', A', B') *with*

$$CC_{\mathbb{R}}(R', A', B') \leq CC_{\mathbb{R}}(R, A, B) \perp 1 \ .$$

2. *if* $\ell \geq 1$ *and for some* $1 \leq j \leq k$, $AVDEG_j(A) < 8m^{\frac{13}{14}}$, *then there is an* $(\alpha + 3 \perp \frac{\log m}{14}, \beta+1, \ell \perp 1)$*-game* (R', A', B') *with*

$$CC_{\mathbb{R}}(R', A', B') \leq CC_{\mathbb{R}}(R, A, B) \ .$$

To prove Theorem 4.1.2 from the lemma, we show that for every (α, β, ℓ)-game (R, A, B),

$$CC_{\mathbb{R}}(R, A, B) \geq \ell \cdot \left(\frac{\log m}{42} \perp \frac{4}{3}\right) \perp \frac{\alpha+\beta}{3} \ . \tag{4.3}$$

The case $\alpha = \beta = 0$ gives the theorem.

For $\ell = 0$ and $\beta > m^{\frac{1}{7}}$, the inequality (4.3) is trivial, since the right hand side gets negative for large m. We proceed inductively: Let (R, A, B) be an (α, β, ℓ)-game, and assume that (4.3) holds for all (α', β', ℓ')-games with $\ell' \leq \ell$ and $\beta' > \beta$. For sake of contradiction, suppose that $CC_{\mathbb{R}}(R, A, B) < \ell \cdot \left(\frac{\log m}{42} \perp \frac{4}{3}\right) \perp \frac{\alpha+\beta}{3}$. Then either for every $1 \leq j \leq k$, $AVDEG_j(A) \geq 8m^{\frac{13}{14}}$, and Lemma 4.1.6 gives an $(\alpha+2, \beta+1, \ell)$-game (R', A', B') with

$$\begin{aligned} CC_{\mathbb{R}}(R', A', B') &\leq CC_{\mathbb{R}}(R, A, B) \perp 1 \\ &< \ell \cdot \left(\frac{\log m}{42} \perp \frac{4}{3}\right) \perp \frac{(\alpha+2)+(\beta+1)}{3} \ , \end{aligned}$$

or for some $1 \leq j \leq k$, $AVDEG_j(A) < 8m^{\frac{13}{14}}$, then Lemma 4.1.6 gives an $(\alpha + 3 \perp \frac{\log m}{14}, \beta + 1, \ell \perp 1)$-game (R', A', B') with

$$\begin{aligned} CC_{\mathbb{R}}(R', A', B') &< \ell \cdot \left(\frac{\log m}{42} \perp \frac{4}{3}\right) \perp \frac{\alpha+\beta}{3} \\ &= (\ell \perp 1) \cdot \left(\frac{\log m}{42} \perp \frac{4}{3}\right) \perp \frac{(\alpha + 3 \perp \frac{\log m}{14}) + (\beta+1)}{3} \ , \end{aligned}$$

both contradicting the assumption.

Proof of Lemma 4.1.6: For part 1, we first show that $CC_{\mathbb{R}}(R, A, B) > 0$. Assume otherwise, then there is a term z in the DNF tautology F_R defining R that is satisfied for every $(x, y) \in$

$A \times B$. Therefore $y_j(x_j)$ is constant for some $1 \leq j \leq k$. If γ denote the number of possible values of x_j in elements of A, then this implies that $|B| \leq 2^{mk-\gamma}$. On the other hand, $|B| \geq 2^{mk-\beta}$, hence it follows that $\beta \geq \gamma$, which is a contradiction since $\beta \leq m^{\frac{1}{7}}$, whereas $AVDEG_j(A) \geq 8m^{\frac{13}{14}}$ implies $\gamma \geq 8m^{\frac{13}{14}}$.

Let an optimal real communication protocol solving R restricted to $A \times B$ be given. For $a \in A$ and $b \in B$, let ρ_a and σ_b be the real numbers played by I and II in the first round on input a and b, respectively. W.l.o.g. we can assume that these are $|A| + |B|$ pairwise distinct real numbers.

Now consider a $\{0,1\}$-matrix of size $|A| \times |B|$ with columns indexed by the ρ_a and rows indexed by the σ_b, both in increasing order, and where the entry in position (ρ_a, σ_b) is 1 if $\rho_a > \sigma_b$ and 0 if $\rho_a \leq \sigma_b$. Thus this entry determines the outcome of the first round, when these numbers are played. It is now obvious that either the upper right quadrant or the lower left quadrant must form a monochromatic rectangle.

Hence there are $A^\circ \subseteq A$ and $B' \subseteq B$ with $|A^\circ| \geq \frac{1}{2}|A|$ and $|B'| \geq \frac{1}{2}|B|$ such that R restricted to $A^\circ \times B'$ can be solved by a protocol with one round fewer than the original protocol. By Lemma 4.1.4 (4.1), $AVDEG_j(A^\circ) \geq 4m^{\frac{13}{14}}$ for every $1 \leq j \leq k$, hence by Corollary 4.1.3 there is $A' \subseteq A^\circ$ with $|A'| \geq \frac{1}{2}|A^\circ| \geq \frac{1}{4}|A|$ and $Thickness(A') \geq m^{\frac{11}{14}}$. Thus (R, A', B') is an $(\alpha + 2, \beta + 1, \ell)$-game.

For part 2 we proceed like in the proof of the corresponding lemma of [66], with the numbers slightly adjusted. Assume without loss of generality that in k is the coordinate for which $AVDEG_k(A) < 8m^{\frac{13}{14}}$. Let R_0 and R_1 be the restrictions of R in which the k-th coordinate $e_k = y_k(x_k)$ is fixed to 0 and 1, respectively. Obviously, R_0 and R_1 are $DART(m, k \perp 1)$ relations, and therefore at least one of $SC(R_0)$ and $SC(R_1)$ is at least $k \perp 1$. Assume without loss of generality that $SC(R_0) \geq k \perp 1$. We will prove that there are two sets $A' \subseteq [m]^{k-1}$ and $B' \subseteq (\{0,1\}^m)^{k-1}$ such that the following properties hold:

$$|A'| \geq \frac{m^{k-1}}{2^{\alpha+3-\frac{\log m}{14}}} \tag{4.4}$$

$$|B'| \geq \frac{2^{m(k-1)}}{2^{\beta+1}} \tag{4.5}$$

$$Thickness(A') \geq m^{\frac{11}{14}} \tag{4.6}$$

$$CC_{\mathbb{R}}(R_0, A', B') \leq CC_{\mathbb{R}}(R, A, B) \tag{4.7}$$

This means that there is a $(\alpha+3\perp\frac{\log m}{14}, \beta+1, k\perp 1)$-game (R_0, A', B') such that $CC_{\mathbb{R}}(R_0, A', B') \leq CC_{\mathbb{R}}(R, A, B)$ and this proves part 2 of Lemma 4.1.6.

Given any set $U \subset [m]$ consider the sets $A_U \subseteq [m]^{k-1}$ and $B_U \subseteq (\{0,1\}^m)^{k-1}$ associated to the set U by the following definition of [66]:

- $(x_1, \ldots x_{k-1}) \in A_U$ iff there is an $u \in U$ such that $(x_1, \ldots x_{k-1}, u) \in A$;
- $(y_1, \ldots y_{k-1}) \in B_U$ iff there is a $w \in \{0,1\}^m$ such that $w(u) = 0$ for all $u \in U$ and $(y_1, \ldots y_{k-1}, w) \in B$.

The following two claims can be proved exactly as the corresponding Claims of [66] and we omit their proof.

Claim 4.1.1 *For a random set U of size $m^{\frac{5}{14}}$ we have that*

$$Prob_U\left[A_U = A[k]\right] \geq \frac{3}{4}\,.$$

Claim 4.1.2 *For a random set U of size $m^{\frac{5}{14}}$ we have that*

$$Prob_U\left[|B_U| \geq \frac{|B|}{2^{m+1}}\right] \geq \frac{3}{4}\,.$$

Moreover it is immediate to see that the same reduction used in Claim 6.3 of [66] also works for the case of real communication complexity. Therefore we get:

Claim 4.1.3 *For every set $U \subset [m]$*

$$CC_{\mathbb{R}}(R_0, A_U, B_U) \leq CC_{\mathbb{R}}(R, A, B)\,.$$

Take a random set U which with probability greater than $\frac{1}{2}$, satisfies both the properties of Claim 4.1.1 and Claim 4.1.2, and define $A' := A_U$ and $B' := B_U$. This means that with probability at least $\frac{1}{2}$ both $A' = A[k]$ and $|B'| \geq \frac{|B|}{2^{m+1}}$ hold.

Recall that $\frac{|A|}{|A'|} = \frac{|A|}{|A[k]|} = AVDEG_k(A)$ and that, by hypothesis on Part 2 of the lemma $|\,AVDEG_k(A)| \leq 8m^{\frac{13}{14}}$. Therefore we have that

$$|A'| \geq \frac{|A|}{8m^{\frac{13}{14}}} \geq \frac{m^k}{2^\alpha 8m^{\frac{13}{14}}} = \frac{m^{k-1}}{2^{\alpha+3-\frac{\log m}{14}}} .$$

This proves (4.4). For (4.5) observe that by Claim 4.1.2 we have

$$|B'| \geq \frac{|B|}{2^{m+1}} \geq \frac{2^{mk}}{2^\beta 2^{m+1}} = \frac{2^{m(k-1)}}{2^{\beta+1}} .$$

The property (4.6) follows directly from Lemma 4.1.4 (4.2), and finally (4.7) follows from Claim 4.1.3. $\square$

4.2 Separation between tree-like and dag-like versions of Resolution and Cutting Planes

Cutting Planes refutations are linked to monotone real circuits by the following interpolation theorem due to Pudlák:

Theorem 4.2.1 (Pudlák [63]) *Let $\vec{p}, \vec{q}, \vec{r}$ be disjoint vectors of variables, and let $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$ be sets of inequalities in the indicated variables such that the variables $\vec{p}$ either have only nonnegative coefficients in $A(\vec{p}, \vec{q})$ or have only non-positive coefficients in $B(\vec{p}, \vec{r})$.*

Suppose there is a CP refutation R of $A(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$. Then there is a monotone real circuit $C(\vec{p})$ of size $O(|R|)$ such that for any vector $\vec{a} \in \{0,1\}^{|\vec{p}|}$

$$C(\vec{a}) = 0 \quad \rightarrow \quad A(\vec{a}, \vec{q}) \text{ is unsatisfiable}$$
$$C(\vec{a}) = 1 \quad \rightarrow \quad B(\vec{a}, \vec{r}) \text{ is unsatisfiable}$$

Furthermore, if R is tree-like, then $C(\vec{p})$ is a monotone real formula.

We now define an unsatisfiable set of clauses related to GEN_n. The variables $p_{a,b,c}$ for $a, b, c \in [n]$ represent the input to GEN_n. Variables $q_{i,j,a}$ for $(i, j) \in Pyr_d$ and $a \in [n]$ encode a pyramid, where the element a is assigned to the position (i, j) by a certain mapping $m : Pyr_d \to [n]$ (cf. Corollary 4.1.2). Finally the variables r_a for $a \in [n]$ represent a coloring of the elements by $0, 1$ such that 1 is colored 0, n is colored 1 and the elements colored 0 are closed under generation. The set $Gen(\vec{p}, \vec{q})$ is given by (4.8) - (4.11), and $Col(\vec{p}, \vec{r})$ by (4.12) - (4.14).

$$\bigvee_{1 \le a \le n} q_{i,j,a} \qquad \text{for } (i, j) \in Pyr_d \tag{4.8}$$

$$\bar{q}_{d,j,a} \vee p_{1,1,a} \qquad \text{for } 1 \le j \le d \text{ and } a \in [n] \tag{4.9}$$

$$\bar{q}_{1,1,a} \vee p_{a,a,n} \qquad \text{for } a \in [n] \tag{4.10}$$

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee p_{a,b,c} \qquad \text{for } (i, j) \in Pyr_{d-1} \text{ and } a, b, c \in [n] \tag{4.11}$$

$$\bar{p}_{1,1,a} \vee \bar{r}_a \qquad \text{for } a \in [n] \tag{4.12}$$

$$\bar{p}_{a,a,n} \vee r_a \qquad \text{for } a \in [n] \tag{4.13}$$

$$r_a \vee r_b \vee \bar{p}_{a,b,c} \vee \bar{r}_c \qquad \text{for } a, b, c \in [n] \tag{4.14}$$

If $Gen(\vec{t}, \vec{q})$ is satisfiable for a fixed vector $\vec{t} \in \{0, 1\}^{n^3}$, then n is generated in a depth-d pyramidal fashion, and if $Col(\vec{t}, \vec{r})$ is satisfiable, then $\text{GEN}(\vec{t}) = 0$. Since the variables $\vec{p}$ occur only positively in $Gen(\vec{p}, \vec{q})$ and only negatively in $Col(\vec{p}, \vec{r})$, Theorem 4.2.1 is applicable, and the formula obtained from this application satisfies the conditions of Corollary 4.1.2. Hence we can conclude:

Theorem 4.2.2 *For some $\epsilon > 0$, every tree-like CP refutation of the clauses $Gen(\vec{p}, \vec{q}) \cup Col(\vec{p}, \vec{r})$ has to be of size $2^{\Omega(n^\epsilon)}$.*

On the other hand, there are polynomial size dag-like resolution refutations of these clauses.

Theorem 4.2.3 *There are (dag-like) resolution refutations of size $n^{O(1)}$ of the clauses $Gen(\vec{p}, \vec{q}) \cup Col(\vec{p}, \vec{r})$.*

Proof: First we resolve clauses (4.9) and (4.12) to get

$$\bar{q}_{d,j,c} \vee \bar{r}_c \tag{4.15}$$

for $1 \leq j \leq d$ and $1 \leq c \leq n$.

Now we want to derive $\bar{q}_{i,j,c} \vee \bar{r}_c$ for every $(i,j) \in Pyr_d$ and $1 \leq c \leq n$, by induction on i downward from d to 1. The induction base is just (4.15).

Now by induction we have

$$\bar{q}_{i+1,j,a} \vee \bar{r}_a \quad \text{and} \quad \bar{q}_{i+1,j+1,b} \vee \bar{r}_b\ ,$$

we resolve them against (4.14) to get $\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{p}_{a,b,c} \vee \bar{r}_c$ for $1 \leq a,b,c \leq n$ and then resolve them against (4.11) and get

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee \bar{r}_c$$

for every $1 \leq a,b \leq n$. All of these are then resolved against two instances of (4.8), and we get the desired $\bar{q}_{i,j,c} \vee \bar{r}_c$ for every $1 \leq c \leq n$.

Finally, we have in particular $\bar{q}_{1,1,a} \vee \bar{r}_a$ for every $1 \leq c \leq n$. We resolve them with (4.13) and get $\bar{q}_{1,1,a} \vee \bar{p}_{a,a,n}$ for every $1 \leq a \leq n$. These are resolved with (4.10) to get $\bar{q}_{1,1,a}$ for every $1 \leq a \leq n$. Finally, this clause is resolved with another instance of (4.10) (the one with $i = j = 1$) to get the empty clause. □

It is easy to check that the above refutation is an N-resolution refutation. The following corollary is an easy consequence of the previous theorems and known simulation results.

Corollary 4.2.1 *The clauses $Gen(\vec{p},\vec{q}) \cup Col(\vec{p},\vec{r})$ exponentially separate tree-like resolution from dag-like Resolution and (dag-like) N-resolution as well as tree-like Cutting Planes from dag-like Cutting Planes.*

The resolution refutation of $Gen(\vec{p},\vec{q}) \cup Col(\vec{p},\vec{r})$ that appears in the proof of Theorem 4.2.3 is not regular. We do not know whether $Gen(\vec{p},\vec{q}) \cup Col(\vec{p},\vec{r})$ has polynomial size regular resolution refutations. To obtain a separation between tree-like Resolution and Regular Resolution we will modify the clauses $Col(\vec{p},\vec{r})$.

4.2.1 Separation of tree-like CP from regular resolution

The clauses $Col(\vec{p},\vec{r})$ are modified (and the modification called $RCol(\vec{p},\vec{r})$), so that $Gen(\vec{p},\vec{q}) \cup RCol(\vec{p},\vec{r})$ allow small regular resolutions, but in such a way that the lower bound proof still applies. We replace the variables r_a by $r_{a,i,D}$ for $a \in [n]$, $1 \le i \le d$ and $D \in \{L,R\}$, giving the coloring of element a, with auxiliary indices i being a row in the pyramid and D distinguishing whether an element is used as a left or right predecessor in the generation process.

The set $RCol(\vec{p},\vec{r})$ is defined as follows:

$$\bar{p}_{1,1,a} \vee \bar{r}_{a,d,D} \qquad \text{for } a \in [n] \text{ and } D \in \{L,R\} \tag{4.16}$$

$$\bar{p}_{a,a,n} \vee r_{a,1,D} \qquad \text{for } a \in [n] \text{ and } D \in \{L,R\} \tag{4.17}$$

$$r_{a,i+1,L} \vee r_{b,i+1,R} \vee \bar{p}_{a,b,c} \vee \bar{r}_{c,i,D} \qquad \text{for } i < d,\ a,b,c \in [n] \text{ and } D \in \{L,R\} \tag{4.18}$$

$$\bar{r}_{a,i,D} \vee r_{a,i,\bar{D}} \qquad \text{for } 1 \le i \le d \text{ and } D \in \{L,R\} \tag{4.19}$$

$$\bar{r}_{a,i,D} \vee r_{a,j,D} \qquad \text{for } 1 \le i,j \le d \text{ and } D \in \{L,R\} \tag{4.20}$$

Due to the clauses (4.19) and (4.20), the variables $r_{a,i,D}$ are equivalent for all values of the auxiliary indices i, D. Hence a satisfying assignment for $RCol(\vec{p},\vec{r})$ still codes a coloring of $[n]$ such that elements a with $1,1 \vdash a$ are colored 0, the elements b with $b,b \vdash n$ are colored 1, and the 0-colored elements are closed under generation. Hence if $RCol(\vec{t},\vec{r})$ is satisfiable, then $\textsc{Gen}(\vec{t}) = 0$.

Hence any interpolant for the clauses $Gen(\vec{p},\vec{q}) \cup RCol(\vec{p},\vec{r})$ satisfies the assumptions of Corollary 4.1.2, and we can conclude

Theorem 4.2.4 *Tree-like CP refutations of the clauses $Gen(\vec{p},\vec{q}) \cup RCol(\vec{p},\vec{r})$ have to be of size $2^{\Omega(n^\epsilon)}$.*

On the other hand, we have the following upper bound on (dag-like) regular resolution refutations of these clauses:

Theorem 4.2.5 *There are (dag-like) regular resolution refutations of the clauses $Gen(\vec{p},\vec{q}) \cup RCol(\vec{p},\vec{r})$ of size $n^{O(1)}$.*

Proof: First we resolve clauses (4.9) and (4.16) to get

$$\bar{q}_{d,j,a} \vee \bar{r}_{a,d,D} \tag{4.21}$$

for $1 \leq j \leq d$, $1 \leq a \leq n$ and $D \in \{L, R\}$. Next we resolve (4.10) and (4.17) to get

$$\bar{q}_{1,1,a} \vee r_{a,1,D} \tag{4.22}$$

for $1 \leq a \leq n$ and $D \in \{L, R\}$. Finally, from (4.11) and (4.18) we obtain

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee r_{a,i+1,L} \vee r_{b,i+1,R} \vee \bar{r}_{c,i,D} \tag{4.23}$$

for $1 \leq j \leq i < d$, $1 \leq a, b, c \leq n$ and $D \in \{L, R\}$.

Now we want to derive $\bar{q}_{i,j,a} \vee \bar{r}_{a,i,D}$ for every $(i,j) \in Pyr_d$, $1 \leq a \leq n$ and $D \in \{L, R\}$, by induction on i downward from d to 1. The induction base is just (4.21).

For the inductive step, resolve (4.23) against the clauses

$$\bar{q}_{i+1,j,a} \vee \bar{r}_{a,i+1,L} \quad \text{and} \quad \bar{q}_{i+1,j+1,b} \vee \bar{r}_{b,i+1,R}\ ,$$

which we have by induction, to give

$$\bar{q}_{i+1,j,a} \vee \bar{q}_{i+1,j+1,b} \vee \bar{q}_{i,j,c} \vee \bar{r}_{c,i,D}$$

for every $1 \leq a, b \leq n$.

All of these are then resolved against two instances of (4.8), and we get the desired $\bar{q}_{i,j,c} \vee \bar{r}_{c,i,D}$.

Finally, we have in particular $\bar{q}_{1,1,a} \vee \bar{r}_{a,1,L}$, which we resolve against (4.22) to get $\bar{q}_{1,1,a}$ for every $a \leq n$. From these and an instance of (4.8) we get the empty clause. □

Note that the refutation given in the proof of Theorem 4.2.5 is actually a Davis-Putnam refutation: It respects the following elimination order

$$\begin{array}{l}
p_{1,1,1} \cdots p_{n,n,n} \\
r_{1,d,L} \;\; r_{1,d,R} \;\; \cdots \;\; r_{n,d,L} \;\; r_{n,d,R} \\
q_{1,d,1} \cdots q_{1,d,n} \cdots q_{d,d,1} \cdots q_{d,d,n} \\
r_{1,d-1,L} \cdots r_{n,d-1,R} \;\; q_{1,d-1,1} \cdots q_{d-1,d-1,n} \\
\vdots \\
r_{1,1,L} \;\; r_{1,1,R} \;\; q_{1,1,1} \cdots q_{1,1,n} \,.
\end{array}$$

Corollary 4.2.2 *The clauses $Gen(\vec{p},\vec{q}) \cup RCol(\vec{p},\vec{r})$ exponentially separate the following proof systems: Tree-like Resolution from Regular and Davis-Putnam Resolution.*

4.3 Lower bound for Davis-Putnam resolutions

Goerdt [38] showed that Davis-Putnam resolution is strictly weaker than unrestricted resolution, by giving a superpolynomial lower bound (of the order $\Omega(n^{\log\log n})$) for Davis-Putnam resolutions of a certain family of clauses, that has polynomial size unrestricted resolution refutations. In this section we improve this separation to exponential, in fact, we give an exponential separation of Davis-Putnam resolution from N-resolution.

To simplify the exposition, we apply the method of [38] to the clauses based on *st*-connectivity that were used by Clote and Setzer [30]. These clauses are formed from variables $p_{\{i,j\}}$ for $0 \leq i,j \leq n+1$ which represent the edges of an undirected graph on $n+2$ nodes, with two distinguished nodes $s = 0$ and $t = n+1$. Variables $q_{i,j}$ for $0 \leq i,j \leq n+1$ code a sequence of nodes of length $n+2$. i.e. $q_{i,j}$ indicates whether j is in the ith position of the sequence. The set of clauses $A(\vec{p},\vec{q})$, expressing that this sequence forms a path from s to t, is given by

$$\begin{array}{ll}
q_{0,s}, \qquad q_{n+1,t} & \\
\bar{q}_{i,j} \vee \bar{q}_{i,k} & \text{for } 0 \leq i \leq n+1 \text{ and } 0 \leq j < k \leq n+1 \\
q_{i,1} \vee \ldots \vee q_{i,n+1} & \text{for } 1 \leq i \leq n \\
\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}} & \text{for } 0 \leq i < n+1 \text{ and } j,k \leq n+1 \text{ with } j \neq k\ .
\end{array}$$

Variables r_i code a coloring of the graph, and the set of clauses $B(\vec{p},\vec{q})$, expressing that s and t have different colors and adjacent nodes have the same color, is given by

$$\bar{r}_s\ , \qquad r_t\ , \qquad r_i \vee \bar{p}_{\{i,j\}} \vee \bar{r}_j \quad \text{for } i,j \leq n+1 \text{ with } i \neq j\ .$$

We shall modify the clauses $A(\vec{p},\vec{q})$ in such a way as to make small Davis-Putnam resolution refutations impossible, while still allowing for small unrestricted resolutions. The lower bound is then proved by a bottleneck counting argument similar to that used in [38], which is based on the original argument of [41].

The set $DP(\vec{p},\vec{q})$ is obtained from $A(\vec{p},\vec{q})$ by adding additional literals to some of the clauses. The clauses $\bar{q}_{i-1,j} \vee \bar{q}_{i,k} \vee p_{\{j,k\}}$ for $1 \leq i \leq \frac{n}{2}$ and $k \leq \frac{n}{4}$ are replaced by

$$\bar{q}_{i',\ell} \vee \bar{q}_{i-1,j} \vee \bar{q}_{i,k} \vee p_{\{j,k\}}$$

for every $\ell \in [n+1]$, where $i' := \frac{n}{2} + 2(k \perp 1) + 1$, and the clauses $\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}}$ for $\frac{n}{2} < i \leq n$ and $j \leq \frac{n}{4}$ are replaced by

$$\bar{q}_{i',\ell} \vee \bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}}$$

for every $\ell \in [n+1]$, where $i' := 2j$. All other clauses remain unchanged.

Theorem 4.3.1 *The clauses $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ have N-resolution refutations of size $n^{O(1)}$.*

Proof: We give an N-resolution refutation of $A(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ from [30] and then show how to adapt that refutation to $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$. By induction on i, we derive the clauses $\bar{q}_{i,j} \vee \bar{r}_j$ for $1 \leq i \leq n+1$. The base case $i = 1$ is done as follows. From $\bar{r}_0$ and $r_0 \vee \bar{p}_{\{0,k\}} \vee \bar{r}_k$

we get $\bar{r}_k \vee \bar{p}_{\{0,k\}}$, which is resolved with $\bar{q}_{0,0} \vee \bar{q}_{1,k} \vee p_{\{0,k\}}$ to produce $\bar{q}_{0,0} \vee \bar{q}_{1,k} \vee \bar{r}_k$, which is then resolved with $q_{0,0}$ to give $\bar{q}_{1,k} \vee \bar{r}_k$ as desired.

For the induction step, we need to derive auxiliary formulas. From $\bar{q}_{i,j} \vee \bar{r}_j$ and $r_j \vee \bar{p}_{\{j,k\}} \vee \bar{r}_k$, we get $\bar{q}_{i,j} \vee \bar{p}_{\{j,k\}} \vee \bar{r}_k$, which is resolved with $\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}}$ giving $\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee \bar{r}_k$. If we resolve $\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee \bar{r}_k$ for all j, $0 \le j \le n+1$ with $q_{i,1} \vee \ldots \vee q_{i,n+1}$ we get $\bar{q}_{i+1,k} \vee \bar{r}_k$. Eventually we get $\bar{q}_{n+1,k} \vee \bar{r}_k$, in particular $\bar{q}_{n+1,n+1} \vee \bar{r}_{n+1}$, and we get the empty clause by resolving it with r_{n+1} and $q_{n+1,n+1}$.

The only difference in the case of the clauses $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ is that sometimes instead of resolving $\bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}}$ with some clause φ to get φ', we will instead have to resolve φ with $\bar{q}_{i',\ell} \vee \bar{q}_{i-1,j} \vee \bar{q}_{i,k} \vee p_{\{j,k\}}$ (or $\bar{q}_{i',\ell} \vee \bar{q}_{i,j} \vee \bar{q}_{i+1,k} \vee p_{\{j,k\}}$) for $1 \le \ell \le n+1$. Now we solve the $n+1$ resulting resolvents (that have only negative literals) with the corresponding clause $q_{i',1} \vee \ldots \vee q_{i',n+1}$ to get φ'. □

In particular, there are polynomial size unrestricted resolution refutations of these clauses.

Definition: A *critical assignment* α is given by the following:

- a coloring col_α such that $col_\alpha(s) = 0$ and $col_\alpha(t) = 1$. The values $\alpha(r_a)$ are assigned according to $col_\alpha(a)$.
- a graph G_α on $\{0, \ldots, n+1\}$ such that no two adjacent edges in G_α have different colors. Values $\alpha(p_{\{i,j\}})$ are assigned according to G_α.
- an index $i_\alpha \in [n]$ such that $\alpha(q_{i_\alpha,k}) = 0$ for all $k \in [n+1]$.
- a mapping $m_\alpha : [n] \setminus \{i_\alpha\} \to [n+1]$, which we extend by defining $m_\alpha(0) = s$ and $m_\alpha(n+1) = t$, such that if $m_\alpha(i)$ and $m_\alpha(i+1)$ are both defined and distinct, then $\{m_\alpha(i), m_\alpha(i+1)\}$ is an edge in G_α. Then we set $\alpha(q_{i,m_\alpha(i)}) = 1$ and $\alpha(q_{i,j}) = 0$ for all $j \ne m_\alpha(i)$, for every $i \ne i_\alpha$.

Obviously, a critical assignment satisfies all clauses from $B(\vec{p},\vec{r})$, and all clauses from $DP(\vec{p},\vec{q})$ except for $\bigvee_{j\in[n+1]} q_{i_\alpha,j}$.

Theorem 4.3.2 *For $n \geq 32$, every Davis-Putnam resolution refutation of the clauses $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ contains at least $2^{\frac{n}{8}}$ clauses.*

Proof: For sake of simplicity, let n be divisible by 8. Let an elimination order $\langle x_1, \ldots, x_N \rangle$ be given, where N is the number of variables, and a Davis-Putnam refutation R of $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ respecting the elimination order.

For $i \in [n]$ and $s \leq N$, let $S(i,s) := \left\{ j \leq \frac{n}{4} \, ; \, q_{i,j} \in \{x_1, \ldots, x_s\} \right\}$. Let i_0 denote the unique index such that there is an index $s_0 \leq N$ with $|S(i_0, s_0)| = \frac{n}{8}$, and for all $i \neq i_0$, $|S(i, s_0)| < \frac{n}{8}$. In other words, i_0 is the first index for which $\frac{n}{8}$ variables $q_{i_0,j}$ with $j \leq \frac{n}{4}$ are eliminated.

Let $S(i_0, s_0)$ be $\{j_1, \ldots, j_{\frac{n}{8}}\}$. For each $1 \leq k \leq \frac{n}{8}$, let $i_k := \frac{n}{2} + 2(j_k \perp 1) + 1$ if $i_0 \leq \frac{n}{2}$, and $i_k := 2j_k$ if $i_0 > \frac{n}{2}$, and define

$$R_k := [\frac{n}{4}] \setminus S(i_k, s_0) ,$$

i.e., R_k is the set of those $j \leq \frac{n}{4}$ for which $q_{i_k,j}$ is eliminated later than any q_{i_0,j_ℓ} for $1 \leq \ell \leq \frac{n}{8}$. Note that $|R_k| \geq \frac{n}{8}$ by definition of i_0.

A critical assignment α is *0-critical*, if $i_\alpha = i_0$ and $m_\alpha(i_k) \in R_k$ for $1 \leq k \leq \frac{n}{8}$, and furthermore the following conditions hold

- if $i_0 \leq \frac{n}{2}$, then $\{m_\alpha(i_0 \perp 1), j_k\}$ is not an edge, but $\{j_k, m_\alpha(i_0 + 1)\}$ is an edge in G_α
- if $i_0 > \frac{n}{2}$, then $\{m_\alpha(i_0 + 1), j_k\}$ is not an edge, but $\{j_k, m_\alpha(i_0 \perp 1)\}$ is an edge in G_α

for every $1 \leq k \leq \frac{n}{8}$. The next lemma shows that there are many 0-critical assignments.

Lemma 4.3.1 *For every choice of pairwise distinct values $b_1, \ldots, b_{\frac{n}{8}}$ with $b_k \in R_k$, there is a 0-critical assignment α with $m_\alpha(i_k) = b_k$ for $1 \leq k \leq \frac{n}{8}$ (the definition of the i_k's comes from the clauses $DP(\vec{p},\vec{q})$).*

Proof: The assignment α is constructed as follows:

- For every index i such that $m_\alpha(i)$ is not defined yet, i.e. $i \neq i_0, i_1, \ldots, i_{\frac{n}{8}}$, assign a value $m_\alpha(i) \neq s, t$ arbitrarily, such that

  - if $i_0 \leq \frac{n}{2}$, then $m_\alpha(i_0 \perp 1)$ does not occur at any other index, and $m_\alpha(i_0 \perp 2) \neq j_k$ for $1 \leq k \leq \frac{n}{8}$.
  - if $i_0 > \frac{n}{2}$, then $m_\alpha(i_0 + 1)$ does not occur at any other index, and $m_\alpha(i_0 + 2) \neq j_k$ for $1 \leq k \leq \frac{n}{8}$,

  and no value occurs both as $m_\alpha(i)$ for some $i < i_0$ and $m_\alpha(i')$ for some $i' > i_0$.

- Put every edge that occurs in this path, i.e. the edges $\{m_\alpha(i), m_\alpha(i+1)\}$ for all $0 \leq i \leq n$ such that $m_\alpha(i)$ and $m_\alpha(i+1)$ are both defined and distinct, into G_α, plus the edges $\{j_k, m_\alpha(i_0 \pm 1)\}$ that are required by the definition of 0-critical.

- Color every $m_\alpha(i)$ with $i < i_0$ by 0, and every other element by 1.

It is obvious from the construction that α is 0-critical. Note that the edges required by the definition of a 0-critical assignment do not connect nodes of different color, since if $i_0 \leq \frac{n}{2}$. then $i_k > i_0$, and if $i_0 > \frac{n}{2}$, then $i_k < i_0$, for every $k \leq \frac{n}{8}$. □

Now we map 0-critical assignments to certain clauses in the proof. For a 0-critical assignment α, let C_α be the first clause in R such that α does not satisfy C_α, and $\{\, j \,;\, q_{i_0,j} \text{ occurs in } C_\alpha \,\} = [n+1] \setminus \{j_1, \ldots, j_{\frac{n}{8}}\}$. This clause exists because α determines a path through R from $\bigvee_{j \in [n+1]} q_{i_0,j}$ to the empty clause, such that α does not satisfy any clause on this path. The variables $q_{i_0,j}$ with $j \leq \frac{n}{4}$ are eliminated along that path, and $q_{i_0,j_1}, \ldots q_{i_0,j_{\frac{n}{8}}}$ are the first among them in the elimination order.

The following lemma shows that the clauses C_α have a certain complexity, which implies that the mapping $\alpha \mapsto C_\alpha$ does not map too many 0-critical assignments to the same clause.

Lemma 4.3.2 *Let α be a 0-critical assignment and $\ell_k := m_\alpha(i_k)$. Then for every $1 \leq k \leq \frac{n}{8}$, the literal $\bar{q}_{i_k,\ell_k}$ occurs in C_α.*

Proof: Let α' be the assignment defined by $\alpha'(q_{i_0,j_k}) := 1$ and $\alpha'(x) := \alpha(x)$ for all other variables x. As q_{i_0,j_k} does not occur in C_α, α' does not satisfy C_α either. If $i_0 \leq \frac{n}{2}$, then the only clause from $DP(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$ not satisfied by α' is

$$\bar{q}_{i_k,\ell_k} \vee \bar{q}_{i_0-1,h} \vee \bar{q}_{i_0,j_k} \vee p_{\{h,j_k\}}$$

where $h := m_\alpha(i_0 \perp 1)$. Recall that by the definition of 0-critical truth assignment, $\{h, j_k\}$ is not an edge. If on the other hand $i_0 > \frac{n}{2}$, then the only clause from $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ not satisfied by α' is

$$\bar{q}_{i_k,\ell_k} \vee \bar{q}_{i_0,j_k} \vee \bar{q}_{i_0+1,h'} \vee p_{\{j_k,h'\}}$$

where $h' := m_\alpha(i_0 + 1)$. Again $\{j_k, h'\}$ is not an edge by the definition of 0-critical truth assignment.

In both cases there is a path through R leading from the clause in question to C_α, such that α' does not satisfy any clause on that path. The variable that is eliminated in the last inference on that path must be one of the q_{i_0,j_r} for $1 \leq r \leq \frac{n}{8}$ by the definition of C_α. Since $\ell_k \in R_k$, the variable q_{i_k,ℓ_k} is later in the elimination order, so it cannot have been eliminated on that path. Therefore $\bar{q}_{i_k,\ell_k}$ still occurs in C_α. □

Now let α, β be two 0-critical assignments such that $\ell_k := m_\alpha(i_k) \neq m_\beta(i_k)$ for some $1 \leq k \leq \frac{n}{8}$, so that $\beta(q_{i_k,\ell_k}) = 0$. By Lemma 4.3.2, the literal $\bar{q}_{i_k,\ell_k}$ occurs in C_α, therefore β satisfies C_α, and hence $C_\beta \neq C_\alpha$.

By Lemma 4.3.1, there are at least $\frac{n}{8}!$ 0-critical assignments that differ in the values $m_\alpha(i_k)$. Thus R contains at least $\frac{n}{8}! \geq (\frac{n}{8e})^{(\frac{n}{8})}$ different clauses of the form C_α, which is at least $2^{(\frac{n}{8})}$ for $n \geq 32$. □

The following corollary is a direct consequence of Theorems 4.3.2 and 4.3.1.

Corollary 4.3.1 *The clauses $DP(\vec{p},\vec{q}) \cup B(\vec{p},\vec{r})$ exponentially separate Davis-Putnam Resolution from unrestricted Resolution and N-resolution.*

A modification similar to the one producing $DP(\vec{p},\vec{q})$ from $A(\vec{p},\vec{q})$ can also be applied to the clauses $Gen(\vec{p},\vec{q})$, yielding a set $DPGen(\vec{p},\vec{q})$. Then for the clauses $DPGen(\vec{p},\vec{q}) \cup Col(\vec{p},\vec{r})$, an exponential lower bound for Davis-Putnam resolutions can be proved by the method of Theorem 4.3.2 (this was presented in the conference version [14] of this paper), and the N-resolution proofs of Theorem 4.2.3 can also be modified for these clauses. Thus the clauses $DPGen(\vec{p},\vec{q}) \cup Col(\vec{p},\vec{r})$ exponentially separate Davis-Putnam from negative resolution as well.

Chapter 5

Resolution and Polynomial Calculus

In [10] Ben-Sasson and Wigderson, based on previous work of Haken and Beame and Pitassi ([41, 6]), defined as a complexity measure for resolution refutations the *width*, that is the maximal number of literals in any clause of the refutation. The importance of considering this measure is twofold. On one side they were able to give a general relationship between the width and the length of a refutation, reducing the problem of giving lower bounds on the length to that of giving lower bounds on the width. The width-size relation can be stated as follows: If F, an unsatisfiable formula over n variables, has a resolution refutation of size S, then it has a resolution refutation of width $O(\sqrt{n \log S})$. Through this relationship they obtained a unified method to prove most of the previously known lower bounds for Resolution. On the other side they made explicit a new simple proof-search algorithm based on searching for clauses of increasing size. This algorithm works in time $T(n) = n^{O(w)}$ where w is the minimal width of any refutation of F.

The trade-off of [10] shows that for k-CNF, with k a constant, if we can give a width lower bound equal to the number of variables, then we have an exponential lower bound for the size of refuting the formula. An immediate interesting question arising from the work [10] is whether we can improve the size-width trade-off there obtained. That is, can we get a weaker width lower bound (e.g. to the square root of the number of variables), but still obtain an exponential lower bound for the size?

In this Chapter we give a negative answer to this question for the case of unrestricted resolution. In fact we find a k-CNF formula built over $O(n^2)$ variables, MGT_n, having polynomial size unrestricted resolution refutations, but requiring $\Omega(n)$ (the square root of the number of variables) width to be refuted. One of the main consequence of our result is that the proof search algorithm of [10] is not going to be efficient for finding resolution refutations. Combining our result with the size-width trade-off for tree-like resolution it turns out that MGT_n requires exponential size tree-like resolution proofs, and therefore it provides an exponential separation between tree-like and unrestricted resolution (see also [14, 10]). Similar exponential separations are still not known for other restrictions of resolution, e.g. regular. Therefore we study whether for restrictions of resolution, different from tree-like, we can obtain a better trade-off result then that given for unrestricted resolution. We give a negative answer to this question. For restrictions of Resolution such as regular, Davis-Putnam, positive, negative and linear, we show the optimality of the same size-width trade-off given for unrestricted resolution. As for the case of unrestricted resolution this fact implies an exponential separation between tree-like resolution and all the restrictions considered. Similar results were obtained in [14]. The separation of tree-like resolution from Linear resolution is new and previously unknown.

Another important, even if negative, consequence of our result is that the proof search algorithm proposed by [10] is not going to be efficient for finding resolution refutations in any of the restrictions of resolution we consider.

In their work [29] Clegg et al gave an efficient proof search algorithm based on the Groebner basis algorithm for finding Polynomial Calculus refutations.

The Polynomial Calculus is not a so strong proof system. Therefore one expects that the Groebner basis algorithm is not very efficient for finding refutations when compared with resolution. To prove a a similar result we should find a formula F over n variables verifying the following two properties: (1) F admits polynomial size resolution refutations; and (2) the Groebner Basis algorithm must run for an exponential time in n to find a refutation of the standrad translation of F to set of polynomials. The latter result iwould be a direct corollary of showing a degree lower bound of $\Omega(\sqrt{n})$ for the standrad polynomial formulation of the formula F. Moreover this result, if obtained, would prove that the simulation of resolution by Polynomial Calculus, given in [29], is optimal when we start with a polynomial size (in n) resolution refutation (see Section 5.4 for details)

Our formula MGT_n seems to be the appropriate candidate to obtain this separation. Nevertheless none of the known techniques to give degree lower bounds in Polynomial Calculus apply.

We show two ways of obtaining a weaker form of the above result: namely we prove that for a given unsatisfiable CNF with polynomial size resolution refutations, its direct translation to polynomials requires Polynomial Calculus refutations of degree not less than $\Omega(\log n)$. Therefore the simulation of resolution in [29] cannot be better than a quasipolynomial, in the case we start with a polynomial size resolution refutation.

The first, and simpler, way we consider follows from an observation based on two recent works [25, 69] studying the complexity of PHP^m_n, for $m > n$ in resolution and Polynomial Calculus (see Section 5 for details).

The second way is a consequence of showing a Polynomial Calculus degree lower bound for a formula known to have polynomial size resolution refutations. Our degree lower bound proof extends the Polynomial Calculus lower bound technique introduced by Razborov in [69], to a formula obtained as a modification of the pigeon hole principle defined by Goerdt in [39]. It is hence of independent interest since this technique was known to work only for the PHP formula.

In the final section we prove that under a fixed standard translation to polynomials the width based algorithm of [10] cannot have a better performance than the Groebner Basis proof search algorithm of [29]. This is a consequence of a lemma giving a Polynomial Calculus simulation of Resolution.

5.1 Preliminaries on the size width trade-off

In this Chapter we denote by $R \vdash F$ that R is a refutation of F (we use the notation $\vdash_{tl}$ to mean that the proof is tree-like). $|R|$ denote the size of a refutation R, that is is the number of clauses used in R. The size complexity $S(\vdash F)$ of refuting a CNF formula F in resolution, is defined as $\min_{R \vdash F} |R|$.

Following [10] the *width* $w(F)$ of a CNF formula F is defined to be the the number of literals of the largest clauses in F. The *width* $w(R)$ of a refutation R is defined as the size of the greatest clause appearing in R. The width $w(\vdash F)$ of refuting a formula F is defined as $\min_{R \vdash F} w(R)$

The following theorem, proved in [10], gives the trade-off between size and width for the tree-like and the dag-like resolution systems:

Theorem 5.1.1 *([10]) Let F be any unsatisfiable CNF formula over n variables. Then:*

1. $S(\vdash_{tl} F) \geq 2^{(w(\vdash F) - w(F))}$;

2. $S(\vdash F) \geq \exp(\Omega(\frac{(w(\vdash F) - w(F))^2}{|Lit(F)|}))$.

The Polynomial Calculus (PC) is a refutation system for formulas in CNF. We express a CNF formula F as a sequence of polynomials $p_1, = 0, \ldots, p_m = 0$ over some field K. To force 0-1 solutions we always add among the initial polynomials the axioms $x^2 \perp x = 0$ for all variables x. A PC refutation is a sequence of polynomials ending with $1 = 0$ such that each line in the sequence is either an initial polynomial or is obtained from two previous polynomials in the sequence by the following rules:

1. sum: $\frac{f \quad g}{\alpha f + \beta g}$ for $\alpha, \beta \in K$;

2. product: $\frac{f}{xf}$, for any variable x.

The degree of a refutation is the maximal degree of a polynomial used in the proof. The complexity of a PC refutation is given by its degree.

We define a *standard mapping tr* from formulas in CNF to sets of polynomials in the following way: (1) $tr(x) = 1 \perp x$; (2) $tr(\bar{x}) = x$; (3) $tr(x \vee y) = tr(x) \cdot tr(y)$. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

5.1.1 Graph Tautology and Modified PHP

A binary relation R is *asymmetric* if and only if for all pairs (x, y), $(x, y) \in R \to (y, x) \notin R$ and is *linear* if for all (x, y), either $(x, y) \in R$ or $(y, x) \in R$. A **(strict) ordering** is a transitive and asymmetric relation. A **linear (strict) ordering** is a is strictly order which is also linear.

It is straightforward to note that any strict order over a finite set defines (at least) a minimal element. When we consider the directed graph associated to a strict order, the previous property is equivalent to the following: *each directed acyclic graph, closed under transitive and without loops must have a source node.*

We define a CNF formula, GT_n, expressing the negation the previous property. Let $x_{i,j}$, $\forall i, j \in [n]$, $i \neq j$, a variable whose intended meaning is that (i, j) is a directed edge in a graph over $[n]$. GT_n is then defined by the following clauses over $n(n \perp 1)$ variables:

$$\begin{array}{lll}
(1) & \bar{x}_{i,j} \vee \bar{x}_{j,k} \vee x_{i,k} & i, j, k \in [n], i \neq j \neq k \neq i \\
(2) & \bar{x}_{i,j} \vee \bar{x}_{j,i} & i, j \in [n], i \neq j \\
(3) & \bigvee_{k=1, k \neq j}^{n} x_{k,j} & j \in [n]
\end{array}$$

where the clauses in (1) say that the graph is transitive, those in (2) that is asymmetric and those in (3) that there is no source node.

Krishnamurthy in [57] was the first to consider this formula and to study its complexity for resolution refutations. He conjectured that GT_n required long proofs in resolution. Stalmark in [75] refuted Krishnamurthy's conjecture giving polynomial size unrestricted resolution refutations.

We will use a tautology encoding a modification of the pigeon hole principle defined in [39]. Let n be a natural number of the form 2^k, for some k, and let $m = \log_2 n$. For each

$j = 1, \ldots, m$, let $Part(j)$ be the partition of $[n]$ induced by j the following way:

$Part(j) := \{\{i, i+1, \ldots, i+(2^j \perp 1)\} \mid i = 1, 1+2^j, 1+2\cdot 2^j, \ldots, 1+(\frac{n}{2^j} \perp 1)\cdot 2^j\}$

If we consider $[n]$ as a set of pigeons and $[m]$ a set of holes, then for each hole $j \in [m]$, $Part(j)$ contains sets of pigeons, e.g.

$$\begin{aligned}
&Part(1) = \{\{1,2\},\{3,4\},\ldots\{n \perp 1, n\}\}\\
&Part(2) = \{\{1,2,3,4\},\ldots,\{n \perp 3, n \perp 2, n \perp 1, n\}\}\\
&\vdots\\
&Part(\log_2 n) = \{\{1,2,\ldots,n\}\}
\end{aligned}$$

Consider the following definition:

Definition 5.1.1 *For all $i, i' \in [n]$, i and i' are j-COMPATIBLE if and only if they are in different sets of $Part(j)$.*

Goerdt in [39] considered the following property for n pigeons and $\log_2 n$ holes. If each pigeon is sitting in some hole, then there must exist an hole j and two pigeons i and i' that are not j-compatible sitting in hole j. Our CNF formula $MPHP_n$ expresses the negation of the previous property with the further restriction that each pigeon must sit in exactly one hole.

$$\begin{array}{ll}
\bigvee_{j=1}^{m} x_{i,j} & i \in [n]\\
\bar{x}_{i,j} \vee \bar{x}_{i',j} & j \in [m], i \neq i' \in [n] \text{ , not } j\text{-compatible}\\
\bar{x}_{i,j} \vee \bar{x}_{i,k} & i \in [n], j \neq k \in [m]
\end{array}$$

The clauses defining our $MPHP_n$ subsume the clauses defining the $MPHP_n$ of [39]. Goerdt gave in [39] polynomial size unrestricted refutations for $MPHP_n$.

Theorem 5.1.2 *([39]) There are polynomial size resolution refutations of $MPHP_n$.*

5.2 Optimality of the size-width trade-off

Consider the equation giving the size-width trade-off for the case of unrestricted Resolution:

$$S(\vdash F) \geq \exp(\Omega(\frac{(w(\vdash F) \perp w(F))^2}{|Lit(F)|})).$$

To show that this trade-off is (almost) optimal we have to find an unsatisfiable CNF formula F such that

- $w(F)$ is $O(1)$
- $S(\vdash F)$ is $O(n^{O(1)})$
- $w(\vdash F)$ is $\Omega(\sqrt{|Lit(F)|})$

Note that when F fulfills the three previous conditions, the trade-off formula doesn't give us an exponential lower bound, but $\exp(\Omega(\frac{|Lit(F)|}{|Lit(F)|})$.

We will consider a modification of the formula GT_n, that we call MGT_n, and we show that MGT_n fulfills the above requirements. We will show (see theorem 5.2.2) that polynomial size resolution refutations for MGT_n can be easily obtained from polynomial size resolution refutations of GT_n. Therefore we start by giving a polynomial size resolution refutation for GT_n. This was first given by Stalmark in [75]. Our proof slightly differs from that of Stalmark. However, this difference allows us to show that our refutations respect the further restrictions of being Davis-Putnam (and therefore regular), positive, and linear.

Theorem 5.2.1 *There are polynomial size refutations of GT_n in the following proof systems: (i) dag-like resolution, (ii) Davis-Putnam resolution, (iii) regular resolution, (iv) positive resolution, (v) linear resolution.*

Proof. We start by giving a scheme of the proof. Consider the following abbreviations:

$$C_m(j) := \bigvee_{i=1, i\neq j}^{m} x_{i,j} \text{ for all } j \in [n]$$

$$A(i,j,k) := \bar{x}_{i,j} \vee \bar{x}_{j,k} \vee x_{i,k} \text{ for all } i,j,k \in [n] \;\; i \neq j \neq k \neq i$$

$$B(i,j) := (\bar{x}_{i,j} \vee \bar{x}_{j,i}) \text{ for all } i,j \in [n] \;\; i \neq j$$

The idea of the proof is to obtain clauses of the form $C_m(j)$ from $m = n$ down to $m = 2$ in the following way:

$$\begin{array}{lllll}
C_n(1) & C_n(2) & \dots & C_n(n \perp 1) & C_n(n) \\
C_{n-1}(1) & C_{n-1}(2) & \dots & C_{n-1}(n \perp 1) & \\
\vdots & & & & \\
C_2(1) & C_2(2) & & &
\end{array}$$

For each k, $C_k(1), \dots, C_k(k)$ are obtained in parallel, and for $j = 1, \dots, k \perp 1$, each $C_{k-1}(j)$ is obtained using the clauses $C_k(j)$ and $C_k(k)$ derived in the previous step, and the initial clauses $A(1,k,j), A(2,k,j), \dots, A(k \perp 1, k, j)$ and $B(j,k)$. At the end we easily derive the empty clause from $C_2(1)$, $C_2(2)$ and $B(2,1)$. Now we provide more details of the proof and show that it respects the various restrictions of resolution.

Consider the following abbreviations:

$$\begin{array}{ll}
D^j_{k-1}(i) := C_{k-1}(j) \vee \bar{x}_{i,k} & k \in [n]/\{1\}, i \in [k \perp 1], j \in [n] \\
E^j_{k-1}(i) := (C_{k-1}(j) \vee \bigvee_{\ell=i}^{k-1} x_{\ell,k}) & k \in [n]/\{1\}, i \in [k \perp 1], j \in [n]
\end{array}$$

The proof proceeds by steps going from $m = n$ to $m = 2$. All the clauses $C_n(j)$, for all $j \in [n]$, are initial clauses and therefore derivable. At the $n \perp k + 1$-th step, for each $j = 1, \dots, k \perp 1$, we prove in parallel each of the $C_{k-1}(j)$ in the following way:

(a) Perform in parallel the following resolution steps, each one resolving the variable $x_{k,j}$:

$$\begin{array}{ll}
(1) & \frac{C_k(j) \quad A(1,k,j)}{D^j_{k-1}(1)} \\
(2) & \frac{C_k(j) \quad A(2,k,j)}{D^j_{k-1}(2)} \\
\vdots & \\
(j \perp 1) & \frac{C_k(j) \quad A(j-1,k,j)}{D^j_{k-1}(j-1)} \\
(j) & \frac{C_k(j) \quad B(j,k)}{D^j_{k-1}(j)} \\
(j+1) & \frac{C_k(j) \quad A(j+1,k,j)}{D^j_{k-1}(j+1)} \\
\vdots & \\
(k \perp 1) & \frac{C_k(j) \quad A(k-1,k,j)}{D^j_{k-1}(k-1)}
\end{array}$$

(b) $C_{k-1}(j)$ is then obtained by the following (tree-like) refutation in which we are resolving along the variables $x_{1,k}, x_{2,k}, \ldots, x_{k-1,k}$:

$$\begin{array}{ll}
(1) & \dfrac{C_k(k) \quad D^j_{k-1}(1)}{E^j_{k-1}(1)} \\
(2) & \dfrac{E^j_{k-1}(1) \quad D^j_{k-1}(2)}{E^j_{k-1}(2)} \\
\vdots & \\
(k \perp 1) & \dfrac{E^j_{k-1}(k-1) \quad D^j_{k-1}(k-1)}{C_{k-1}(j)}
\end{array}$$

Such a refutation respects the positive restriction, since at each resolution step one of the premises contains only positive literals. It is also easy to see that the following order of elimination of the variables is respected:

$x_{n,1}, x_{n,2}, \ldots, x_{n,n-1}$

$x_{1,n}, x_{2,n}, \ldots, x_{n-1,n}$

$x_{n-1,1}, x_{n-1,2}, \ldots, x_{n-1,n}$

$x_{1,n-1}, x_{2,n-1}, \ldots, x_{n-2,n-1}$

$\vdots$

$x_{2,1}$

$x_{1,2}$

Therefore the refutation is Davis-Putnam as well as regular. To see that the refutation is Linear observe that the following sequence of clauses defines the order of the linear elimination:

$C_n(n)$,

  $C_n(1), B(n,1), A(2,n,1), \ldots, A(n \perp 1, n, 1)$

    $D^1_n(1), \ldots, D^1_n(n), E^1_n(1), \ldots, E^1_n(n)$,

  $C_n(2), A(1,n,2), B(n,2), \ldots, A(n \perp 1, n, 2)$,

    $D^2_n(1), \ldots, D^2_n(n), E^2_n(1), \ldots, E^2_n(n)$,

  $\vdots$

  $C_n(n \perp 1), A(1, n, n \perp 1), \ldots, A(n \perp 2, n, n \perp 1), B(n, n \perp 1)$

    $D^{n-1}_n(1), \ldots, D^{n-1}_n(n \perp 1), E^{n-1}_n(1), \ldots, E^{n-1}_n(n \perp 1)$,

$C_{n-1}(n \perp 1)$,

$C_{n-1}(1), B(n \perp 1, 1), A(2, n \perp 1, 1), \ldots, A(n \perp 2, n \perp 1, 1)$,

$D^1_{n-1}(1), \ldots, D^1_{n-1}(n \perp 1), E^1_{n-1}(1), \ldots, E^1_{n-1}(n \perp 1)$,

$C_{n-1}(2), A(1, n \perp 1, 2), B(n \perp 1, 2), \ldots, A(n \perp 2, n \perp 1, 2)$,

$D^2_{n-1}(1), \ldots, D^2_{n-1}(n \perp 1), E^2_{n-1}(1), \ldots, E^2_{n-1}(n \perp 1)$

$\vdots$

$C_{n-1}(n \perp 2), \ldots, A(n \perp 3, n \perp 1, n \perp 2), B(n \perp 1, n \perp 2)$

$D^{n-2}_{n-1}(1), \ldots, D^{n-2}_{n-1}(n \perp 1), E^{n-2}_{n-1}(1), \ldots, E^{n-2}_{n-1}(n \perp 1)$,

$C_{n-2}(n \perp 2)$,

$\vdots$

$\{\}$

$\square$

In the above refutation there are clauses of size $O(n)$. We show below that in fact any refutation of GT_n must have clauses of such width. Note however that in GT_n there are initial clauses of size $n \perp 1$ and therefore GT_n does not fulfills the first requirement needed to show the optimality of the size-width trade-off. We consider a modified version of GT_n, MGT_n, such that: (1) $w(MGT_n) \leq 3$; (2) $|Lit(MGT_n)| = 2n^2 \perp n$; (3) from a refutation of GT_n we can easily find a refutation of MGT_n, and (4) $w(\vdash MGT_n) \geq \Omega(n)$.

Consider the clauses with large width in the definition of GT_n:

$$\bigvee_{k=1, k\neq j}^{n} x_{k,j} \quad \text{for } j \in [n]$$

For each $j \in [n]$ we consider n new *extension* variables $y_{0,j}, \ldots y_{j-1,j}, y_{j+1,j}, \ldots y_{n,j}$. MGT_n is defined substituting in the definition of GT_n the previous clauses by the following set of clauses of constant width:

$$\bar{y}_{0,j} \wedge \bigwedge_{i=1, i\neq j}^{n} (y_{i-1,j} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{n,j} \quad \text{for all } j \in [n]$$

MGT_n has then constant initial width, and it is defined over $2n^2 \perp n$ literals. It remains to prove that: (1) there are polynomial size resolution refutations of MGT_n and (2) the minimal width for refuting MGT_n is $\Omega(n)$ (observe that the number of variables in MGT_n is $O(n^2)$). We start by proving that the upper bound holds, even for the restrictions of resolution.

Theorem 5.2.2 *There are polynomial size refutations for the formula MGT_n in any of the following proof systems: (i) Resolution, (ii) Positive Resolution, (iii) Davis-Putnam Resolution, (iv) Regular Resolution, (v) Linear Resolution.*

Proof. The proof follows a standard argument. From the initial clauses of MGT_n

$$\bar{y}_{0,j} \wedge \bigwedge_{i=1, i\neq j}^{n} (y_{i-1,j} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{n,j} \quad \text{for all } j \in [n]$$

we derive the initial clauses of GT_n

$$\bigvee_{k=1, k\neq j}^{n} x_{k,j} \quad \text{for all } j \in [n]$$

resolving along the y variables once at time and in a tree-like fashion. Then we apply the polynomial size proof of GT_n to these new clauses using the other initial clauses. The part of the proof eliminating the y variables is in fact a tree-like proof of size quadratic in n. Since the y variables are always different, the regularity of the proof is preserved. It is also easy to see that the new first part of the proof is a Davis-Putnam resolution since the following order of elimination of the y variables is respected:

$y_{0,1}, \cdots, y_{n,1},$

$y_{0,2}, \cdots, y_{n,2},$

$\vdots$

$y_{0,n}, \cdots, y_{n,n}.$

To obtain a Positive Resolution refutation we only have to take care of eliminating the $y_{j,k}$ variables starting form $y_{j,n}$ for all $j \in [n]$. Finally, to prove that this proof is also a Linear Resolution refutation, consider for $j = 1, \ldots, n$ the following definition:

$$G_j(i) = \begin{cases} y_{n,j} & \text{if } i = n \\ (x_{n,j} \vee x_{n-1,j} \ldots x_{i,j} \vee y_{i-1,j}) & i = 1, \ldots, n \perp 1 \\ C_n(j) & \text{if } i = 0 \end{cases}$$

Then the order of the clauses in the linear resolution of MGT_n is obtained from the order of the linear resolution for GT_n by putting for each $j = 1, \ldots, n$ the sequence of clauses $G_j(n), \ldots, G_j(1)$ just before the clause $C_n(j)$. □

In order to prove the lower bound for the width of refuting MGT_n, we need to introduce the notion of critical truth assignment for the formula MGT_n. We start by giving the definition of critical assignments for GT_n and then extend it to the case of MGT_n. A *linear directed graph* is the graph associated to a strict linear order, i.e. a directed acyclic graph, closed under transitivty, without loops, and in which each two nodes are linked by an edge (see Figure 1).

Definition 5.2.1 *A critical truth assignment α for GT_n is a linear directed graph.*

The idea is that if the variable $x_{i,j}$ correponds to whether there is a directed edge (i, j) in the graph, then such a linear graph falsifies only one among the initial clauses of GT_n. This is because the graph is closed under transitivity, there are no cycles, and every node except for the first one in the line has a predecessor. We call a critical assignment a j-cta if j is the first element in the linear graph. A j-cta falsifies only the initial clause

$$\bigvee_{i \in [n], i \neq j} x_{i,j}$$

Switching from a j-cta to a k-cta is very easy in terms of the linear graph. We put the node k in the first position in the line, and move all nodes before k one position forward. In terms of the adjacency matrix of the graph this means changing the 1's by 0's in column k and the 0 by 1's in row k. The following matrices show how to obtain a 4-cta from a 2-cta

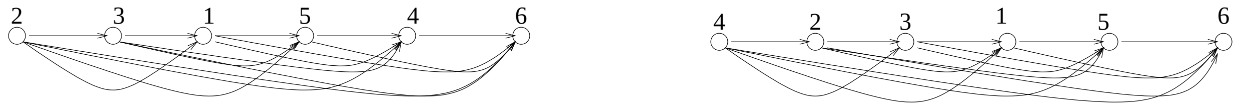

Figure 5.1: Two linear directed graphs giving a 2-cta and a 4-cta for for GT_6.

for GT_6 (see also the previous figure).

2-cta for GT_6

$$\begin{array}{cccccc}
* & 0 & 1 & \mathbf{1} & 1 & 1 \\
1 & * & 0 & \mathbf{1} & 1 & 1 \\
1 & 0 & * & \mathbf{1} & 1 & 1 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & * & \mathbf{0} & 1 \\
0 & 0 & 0 & \mathbf{1} & * & 1 \\
0 & 0 & 0 & 0 & 0 & *
\end{array}$$

4-cta for GT_6

$$\begin{array}{cccccc}
* & 0 & 1 & \mathbf{0} & 1 & 1 \\
1 & * & 0 & \mathbf{0} & 1 & 1 \\
1 & 0 & * & \mathbf{0} & 1 & 1 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & * & \mathbf{1} & 1 \\
0 & 0 & 0 & \mathbf{0} & * & 1 \\
0 & 0 & 0 & 0 & 0 & *
\end{array}$$

Denote with C_j the formula $\bar{y}_{0,j} \wedge \bigwedge_{i=1,i\neq j}^{n}(y_{i-1,j} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{n,j}$ in the definition of MGT_n.

Definition 5.2.2 *A j-cta α for MGT_n is obtained by j-cta for GT_n assigning boolean values to the extension variables y's in such a way that $\alpha \not\models C_j$ and for all $k \neq j$, $\alpha \models C_k$.*

Next Lemma shows that the previous definition is sound, that is we can always extend a j-cta for GT_n to a j-cta for MGT_n.

Lemma 5.2.1 *Any j-cta for GT_n can be extended to a j-cta for MGT_n.*

Proof. Let α be the j-cta for GT_n. β will be the j-cta for MGT_n. α falsifies the initial clause $\bigvee_{k=1,k\neq j}^{n} x_{k,j}$. Hence it falsifies also the formula C_j for any possible assignment to the the variables $y_{i,j}$ for $i = 0, \ldots, n$. To define β we have to specify how to assign values to the $y_{i,k}$ variables for $i = 0, \ldots, n$ and for $k \neq j$, in such a way to satisfy B_k. For any $k \neq j$ let i_k the smallest i greater than 1 such that $\alpha(x_{i,k}) = 1$. β assigns values to the $y_{i,k}$'s the following way: $\beta(y_{i,k}) = 0$ for $i < i_k$, and $\beta(x_{i,k}) = 1$ for $i \geq i_k$. $\Box$

Recall that C_j is the formula $\bar{y}_{0,j} \wedge \bigwedge_{i=1,i\neq j}^{n}(y_{i-1,j} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{n,j}$. Let B_j be the conjunction of the clauses $\bar{x}_{i,j} \vee \bar{x}_{j,i}$ for all $i \in [n], i \neq j$. Consider the formula A_j defined as the conjunction $B_j \wedge C_j$ and let $Vars(j)$ be the set of variables in A_j, that is, in $Vars(j)$ we have all the variables that mention the node j, i.e. the variables $x_{i,j}, \bar{x}_{i,j}, x_{j,i}, \bar{x}_{j,i}$ for all $i \in [n]$, $i \neq j$, and the variables $y_{i,j}, \bar{y}_{i,j}$ for $i \in \{0\} \cup [n]$, $i \neq j$.

Theorem 5.2.3 *Any resolution proof of MGT_n must have a clause of width $\Omega(n)$.*

Proof. For each $I \subseteq [n]$, let A_I be defined as $\bigwedge_{i \in I} A_i$. For any clause C in a resolution proof of MGT_n, let I_C be the minimal $I \subseteq [n]$ such that all critical truth assignments satisfying A_I also satisfy C. For any clause C we define a *measure* $\mu(C)$ as the cardinality of I_C. Observe that for any $i \in [n]$ $\mu(A_i) \leq 1$, moreover $\mu(\{\}) = n$, and μ is obviously subadditive, that is for any step in the resolution, the measure of the conclusion is less than or equal to the sum of the measure of the premises. Therefore in any resolution proof of MGT_n there is a clause, say K, such that $\frac{n}{3} \leq \mu(K) \leq \frac{2n}{3}$. We show that this clause will contain $\geq \frac{n}{6}$ literals. Assume for the sake of contradiction that $|K| < \frac{n}{6}$. Consider the set I_K and notice that, since $|I_K| = \mu(K) \geq \frac{n}{3}$, the following claim holds:

Claim 5.2.1 *There exists at least an $l \in I_K$ such that no variable from $Vars(l)$ belongs to K.*

Proof of the Claim

Each variable $x_{i,j}$ belongs to two different sets $Vars(i)$ and $Vars(j)$ and each variable $y_{s,i}$ belongs to one set $Vars(i)$. Therefore in the worst case K is capturing strictly less than $\frac{2n}{6}$ different sets $Vars(\cdot)$. Since $|I_K| \geq \frac{n}{3}$, then there is at least an index in I_K verifying the claim. $\Box$

Let l be given by the previous Claim. Consider any critical assignment α such that $\alpha(A_l) = 0, \alpha(K) = 0$ and for all $j \in I_K/\{l\}$ $\alpha(A_j) = 1$. This assignment exists by the minimality of I_K and moreover it satisfies all the clauses A_i for $i \in [n]/\{l\}$. Define $J = [n]/I_K$. Since $|J| \geq \frac{n}{3}$ (because $|I_K| \leq \frac{2n}{3}$), applying Claim 5.2.1 to J we deduce that there is at least a $j \in J$ such that no variable from $Vars(j)$ appears in K. We build a j-critical truth assignment β from α such that $\beta(A_i) = 1$ for all $i \in I_K$ and $\beta(K) = 0$, and this leads to a contradiction by the definition of K. β is built the following way: first we change the value of the x's variables in MGT_n, switching the l-cta α to a j-cta. As observed above, this means to change all $x_{i,j}$ such that $\alpha(x_{i,j}) = 1$ to 0 and all the symmetric values $x_{j,i}$ such that $\alpha(x_{j,i}) = 0$ to 1. This first change does not affect the value of K since the variables $x_{i,j}$ and the variables $x_{j,i}$ are in $Vars(j)$ and no variable from $Vars(j)$ appears in K. A this point β is falsifying the formula C_j and *a fortiori* the formula $A_j = B_j \wedge C_j$. The only other initial formula so far affected by our change is A_l. For β to be a j-cta we have to satisfy the formula A_l. Notice however that after the switching, the value of the variable $x_{j,l}$ is 1. Therefore to satisfy A_l we have to satisfy C_l changing the values of the variables $y_{i,l}$ for $i = 0, \dots, n$, as in Lemma 5.2.1. This last change will not affect the value of K since for $i = 0, \dots, n$ the variables $y_{i,l}$ belong to $Vars(l)$ and no variable from $Vars(l)$ appears in K. Notice that the variables $y_{i,s}$ for $s \neq l$ don't need to change value in β. □

The following is an immediate corollary of previous two theorems.

Theorem 5.2.4 *There is a constant width CNF formula F on $O(n^2)$ variables verifying the following two properties: (1) F has polynomial size resolution refutations; (2) any resolution refutation of F contains a clause having width at least $\Omega(n)$.*

5.3 Consequences of the optimality result

Our result has several consequences. First of all the size-width trade-off of [10] for tree-like resolution together with Theorem 5.2.3 give a lower bound of $2^{\Omega(n)}$ for the size of tree-like resolution proofs of MGT_n.

Theorem 5.3.1 *Any tree like resolution proof of MTG_n must have size $\Omega(2^n)$.*

This Theorem allows us to prove that tree-like resolution is exponentially separated from the other restricted systems of resolution we consider. The separation from Linear Resolution is new and previously unknown, the others were only recently obtained in [14, 10].

Theorem 5.3.2 *Tree-like resolution is exponentially separated from unrestricted, Positive, Negative, Regular, Davis-Putnam, and Linear Resolution.*

Proof. For the case of regular, Davis-Putnam, Positive, Linear and unrestricted Resolution, the result is immediate since MGT_n has polynomial size refutation in these systems, but, by previous theorem, require exponential size tree-like refutations. For the case of Negative Resolution, we consider the unsatisfiable formula $\overline{MGT_n}$ in which the $x_{i,j}$ variables are replaced by $\bar{z}_{i,j}$ whose intended meaning is opposite to that of the x variables. It is easy to see that the positive resolution proof for MGT_n is in fact a negative resolution proof for $\overline{MGT_n}$. Moreover the technique to obtain the lower bound for the width can also be applied. We leave to the reader the details of this fact. This means that the shortest tree-like Resolution refutations of $\overline{MGT_n}$ are exponentially long in n. And therefore tree-like Resolution is exponentially separated also from Negative Resolution. □

As we have seen in Section 5.1, the size-width trade-off for tree-like resolution is much better than that for dag-like resolution. This fact, combined with our optimality result for the size-width trade-off for dag-like resolution, allows us to obtain an exponential separation between tree-like and dag-like resolution (Theorem 5.3.2). Therefore an interesting question is to know whether we can also obtain an exponential separation for other restrictions of resolution (e.g. regular, for which the problem is still open), showing better trade-off results than in the unrestricted case. Our next theorem implies that this is not the case for Regular, Positive, Negative, Davis-Putnam and Linear Resolution. In fact we prove that the size-width trade-off for dag-like resolution is optimal even for all these restrictions.

Theorem 5.3.3 *The trade-off*

$$S(\vdash F) \geq \exp(\Omega(\frac{(w(\vdash F) \perp w(F))^2}{|Lit(F)|}))$$

is optimal for the systems of Regular, Positive, Negative, Davis-Putnam and Linear resolution.

Proof. The polynomial size refutations for MGT_n provided in theorem 5.2.2 are also in all the restrictions but Negative Resolution. By Theorem 5.2.3 any resolution refutation of MGT_n (in particular in any of the considered restrictions) must have a clause of size $\Omega(n)$. The result is hence immediate for these restrictions. In the case of Negative Resolution the result follows by an argument similar to that applied in the previous Theorem. $\Box$

A refutation system is **automatizable** if there is an algorithm $\mathcal{A}$ such that, for any unsatisfiable formula F, $\mathcal{A}$ finds a refutation of F in that systems in time polynomial in the size of the shortest proof of F in that system. It is not known if Resolution (or even tree-like Resolution) is automatizable.

Ben-Sasson and Wigderson in [10] considered a simple algorithm, BW, for finding resolution proofs based on the idea of seeking for refutations of minimal width. Let F be a unsatisfiable CNF formula. Consider the following algorithm:

Algorithm BW

```
C:= Clauses of $F$
$w := 0$
While $\Box \notin C$ Do
    $w := w + 1$
    apply resolution rule to clauses in $C$ to derive all possible clauses of width $\leq w$.
    Add the clauses obtained to $C$
End
```

By definition of width it is immediate to see that the possible number of clauses that the above algorithm can produce is bounded by $n^{O(w(\vdash F))}$. Therefore the running time T_{BW} of the algorithm BW is bounded by $n^{O(w(\vdash F))}$. By the size-width trade-off for unrestricted resolution we have that $T_{BW} \leq O(2^{(\sqrt{S(\vdash F)})})$. Our main result (Theorem 5.2.4), implies that BW cannot be used to obtain the automatizability of any of the resolution systems considered. This is because we have a formula, MGT_n, that has polynomial size resolution refutations, but, since the minimal width for refuting MGT_n is $\Omega(n)$, then the algorithm BW will require an exponential number of steps to find a refutation of MGT_n in any of the considered restrictions.

5.4 Degree Lower Bounds for the Modified PHP

In this section we show that any polynomial calculus refutation of the $MPHP_n$ requires degree $\Omega(\log n)$. We will use the same technique of [69, 46]. Recall from Section 5.1 that $m = \log n$, the definition of j-compatible pigeons and the definition of the set $Part(j)$, for all $j \in [m]$. Given $Q_i := 1 \perp \sum_{j \in [m]} x_{i,j}$ we adopt the following polynomial formulation of the $MPHP_n$:

$$\begin{array}{lll}
(1) & Q_i = 0 & i \in [n] \\
(2) & x_{i,j} x_{i,k} = 0 & i \in [n],\ j,k \in [m] \\
(3) & x_{i,j} x_{k,j} = 0 & j \in [m],\ i,k \in [n] \text{ not } j\text{-compatible} \\
(4) & x_{i,j}^2 - x_{i,j} = 0 & i \in [n],\ j \in [m]
\end{array}$$

For a polynomial x which is a product of $x_{i,j}$, let $Pigeons(x, j)$ be the set of i's such that $x_{i,j}$ is a factor in x.

Definition 5.4.1 *T is the set of the monomials $x = x_{i_1,j_1} \ldots x_{i_l,j_l}$ such that all i_k are distinct and for all $j_k \in [m]$ and for all i and i' in $Pigeons(x, j_k)$ i and i' are j_k-compatible. T_d is the set of monomials in T of degree at most d.*

Using the identities (2), (3) and (4) any polynomial can be represented, without increasing its degree, as a linear combination of monomials in T. Therefore any polynomial calculus refutation carried on modulo the ideal I generated by the polynomials (2), (3) and (4), is in the vector space $Span(T)$ generated from the monomials in T. From now on we assume that all the computations are modulo the ideal I.

We want to build a basis B_d for the vector space $Span(T)$ such that the elements of B_d are products of the form $\prod_{i,j} x_{i,j} \prod_i Q_i$. As in [46] (and [69]) the definition of B_d is obtained from a process that maps partial assignments into partial assignments: the pigeon dance. We consider a dummy hole 0, and we represent elements of B_d as partial assignments according to the following definition:

Definition 5.4.2 *A is the set of the partial mappings a from $[n]$ to $[m] \cup \{0\}$ such that for all $i, i' \in [n]$, $i \neq i'$, if $a(i) = a(i') = j \neq 0$ then i and i' are j-compatible.*

Let $A_d := \{a \in A \; : \; |a| \leq d\}$. For $a \in A$ with $a = \{(i_1, j_1), \ldots (i_k, j_k), (i'_1, 0), \ldots , (i'_l, 0)\}$, let $\hat{a}$ denote the restriction $\{(i_1, j_1), \ldots (i_k, j_k)\}$ of a. Any element $a \in A$ defines a polynomial x_a the following way: $x_a = \prod_{a(i)=j, j \neq 0} x_{i,j} \prod_{a(i)=0} Q_i$. Therefore by definition of T any polynomial $x_{\hat{a}}$ associated to $\hat{a} \in A_d$ is in T_d.

Our pigeon dance differs from that of [69, 46] since sometimes a pigeon can be sent to an occupied hole. Consider the following definition:

Definition 5.4.3 *Given $a \in A$, we say that a hole j IS GOOD FOR THE PIGEON i IN a, and we write $j \in Good(i, a)$, if $j > a(i)$ and for all $i' \in a^{-1}(j)$, i and i' are j-compatible.*

Given $a \in A$, our pigeon dance works the following way: starting from the first pigeon in $dom(a)$ we try to move all the pigeons $i \in dom(a)$ into a hole j which is *good* for i in a.

Definition 5.4.4 (Dance) *Let $a \in A$ and consider $dom(a)$. A pigeon dance on a is a sequence of mappings $a_0, a_1, \ldots a_n$ in A with the same domain as a, defined the following way: $a_0 = a$ and for all $0 < t \leq n$, if $a(t)$ is undefined, then $a_t = a_{t-1}$, otherwise*

$$\begin{cases} a_t(j) = a_{t-1}(j) & j \neq t \\ a_t(t) \in Good(t, a_{t-1}) & \end{cases}$$

Definition 5.4.5 (Minimal Dance) *Let $a \in A$ be given and let t be a pigeon index in $[n]$. By $D_t(a)$ we denote a mapping $b \in A$ such that $dom(b) = dom(a)$, and defined as follows:*

$$b(i) = a(i) \qquad i \in dom(a), i \neq t$$
$$b(t) = min_{j \in [m]}[j \in Good(t, a)]$$

If $min_{j\in[m]}[j \in Good(t,a)]$ does not exists, then $D_t(a)$ is undefined. The MINIMAL PIGEON DANCE *$D_{min}(a)$ on a is: $D_{min}(A) = D_n(D_{n-1}(\cdots(D_1(a))\cdots)$*

The minimal dance has two main properties. It can be always defined whenever a dance is defined, and it defines a one-to-one mapping from partial assignments to partial assignments. We show these properties in the following lemmas.

Lemma 5.4.1 *If there exists a dance on a, then there always exists a minimal dance on a.*

Proof. We prove by induction on $t = 1, \ldots, n$ that there is dance $b = b_0, b_1, \ldots, b_n$ where $b_0 = a$ such that its first t steps correspond to the first t steps of the minimal dance on a. The lemma hence follows for $t = n$. Assume to have proved the claim for $t \perp 1$, and let $b = b_0, b_1, \ldots, b_n$ the correct dance having the first $t \perp 1$ steps as in the minimal dance. We show how to build a new correct dance $c = c_0, c_1, \ldots, c_n$ having its first t steps as in the minimal dance.

Let $j_{min} = \min_{j\in[m]}[j \in Good(t, b_{t-1})]$ and suppose $j = b_t(t)$. Observe that since b is a correct dance, then j_{min} always exists and moreover $j_{min} \leq j$. If $j = j_{min}$, then b is making the right choice at the t-th step. In this case we have no need to change b, so we define $c_i = b_i$ for all $i = 0, \ldots, n$. Assume instead that $j_{min} < j$. In this case we define $c = c_0, c_1, \ldots, c_n$ the following way: in the first $t \perp 1$ steps c and b are the same, that is, for all i, $i = 1, \ldots, t \perp 1$, $c_i = b_i$; at the t-th step, c_t is defined by:

$$c_t(i) = \begin{cases} j_{min} & \text{if } i = t \\ b_t(i) & \text{otherwise} \end{cases}$$

The definition of c_i for for $i > t$ is as follows:

$$c_i(j) = c_{i-1}(j) \quad \text{for } j \neq i$$
$$c_i(i) = \begin{cases} j & \text{if } b_i(i) = j_{min} \text{ and } i \text{ and } t \text{ are not } j_{min}\text{-compatible} \\ b_i(i) & \text{otherwise} \end{cases}$$

We have to prove that $c = c_0, c_1, \ldots, c_n$ is a properly defined dance and its first t steps are minimal. Observe that the first $t \perp 1$ steps of c are correct and minimal since they are the same of b. The t-th step is correct and minimal by definition of j_{min}. Therefore it remains to prove that the steps strictly greater than t define a correct dance. By the definition of c_i, for $i > t$, we have to prove that for all $i > t$ $c_i(i) \in Good(i, c_{i-1})$.$\square$

Claim 5.4.1 *For all $i > t$, $c_i(i) \in Good(i, c_{i-1})$.*

Proof. (**of Claim5.4.1**)
By the definition of $c_i(i)$ for $i > t$, it is easy to see that we have to prove that for all $i > t$ such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible, then $j \in Good(i, c_{i-1})$. We obtain the Claim showing that: (1) there is at most one i such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible; and (2) for this i we have that $j \in Good(i, c_{i-1})$.

The first property easily follows since if there exist two different pigeons i and i' both not j_{min}-compatible with t, then i, i' and t are in the same set $D \in Part(j_{min})$. But this is not possible since b is a correct dance and therefore i and i' must be j_{min}-compatible.

For the second point, assume we have an i such that $b_i(i) = j_{min}$ and i and t are not j_{min}-compatible. We prove that i is j-compatible with all elements in $c_{i-1}^{-1}(j)$, which proves that $j \in Good(i, c_{i-1})$. If $c_{i-1}^{-1}(j) = \emptyset$, the result is trivial. Assume, for the sake of contradiction, that there is a $i' \in c_{i-1}^{-1}(j)$, which is not j-compatible with i. Therefore i and i' are in the same set $B \in Part(j)$. Since i is the only pigeon on which we have modified the dance b (except for t), then it follows that i' was already sent to j in b, that is $b_{i'}(i') = j$. Observe that, since i and t are not j_{min}-compatible, then i and t are in the same set $C \in Part(j_{min})$. But since $j_{min} < j$, then $C \subset B$ and therefore $t \in B$. This is a contradiction since b is a correct dance and we cannot have that $b_t(t) = j$ and $b(i) = j$, for two pigeons i and t not j-compatible. $\square$

Lemma 5.4.2 *The minimal dance is a one-to-one mapping.*

Proof. We show that for all $t = 1, \dots, n$, $D_t(\cdot)$ is a 1-1 mapping. The result then follows since the minimal dance is a composition of the D_t mappings. We show that if $D_t(a) = D_t(a')$

then $a = a'$. Suppose $D_t(a) = D_t(a')$. Then $dom(a) = dom(a')$ and $a(i) = a'(i)$ for all $i \in dom(a), i \neq t$. It remains to show that $a(t) = a'(t)$. We show that neither $a(t) < a'(t)$ nor $a'(t) < a(t)$. Suppose the former. We show the following two inequalities:

$$D_t(a)(t) \leq a'(t) \quad a'(t) < D_t(a')(t)$$

This leads to a contradiction since $D_t(a')(t) = D_t(a)(t)$ and by previous two inequalities we have that $D_t(a)(t) < D_t(a)(t)$. The second inequality just follows from the definition of minimal dance. To obtain the first inequality, observe that since for all $i < t$, $a(i) = a'(i)$, then either $Good(t, a) \subseteq Good(t, a')$ or $Good(t, a') \subseteq Good(t, a)$ and moreover $Good(t, a')$ and $Good(t, a)$ share the same minimal element $D_t(a)(t) = D_t(a')(t)$. Since $a'(t) \in Good(t, a')$, then $a'(t) \geq D_t(a)(t)$. The other case $a'(t) < a(t)$ is completely symmetric. □

Consider the following fact:

Fact

If a pigeon dance ends successfully on an $a \in A$, then the polynomial associated to the dance is in T (this is because we are moving to always strictly greater holes and therefore at the end the dummy hole 0 has disappeared).

Lemma 5.4.3 *If $d \leq \frac{\log n}{2}$ and $a \in A_d$, then there exists a dance on a if and only if there exists a dance on $\hat{a}$.*

Proof. If there is a dance for a then obviously there is a dance for $\hat{a}$, so one implication is easy. For the other implication, assume that the number of Q factors in x_a is different from 0, since otherwise there is nothing to prove. Assume that $\hat{a} = \{(i_1, j_1), \ldots, (i_k, j_k)\}$ and that l is the number of Q's factors in a, so that $k + l < d = \frac{\log n}{2}$. Note that after excuting the dance on $\hat{a}$ we remain with at least l holes unused. We will use these unused holes to define a dance on the whole a. That is, if the pigeon i is in $dom(\hat{a})$, then $a(i) = \hat{a}(i)$. If the pigeon $i \in dom(a) \perp dom(\hat{a})$, then we assign one of the unused l holes to move i in $a(i)$. Since these are completely new holes and since $|dom(a)| \perp |dom(\hat{a})| \leq l$, then the dance on a is well defined. □

We can now proceed to the definition of the basis B_d.

Definition 5.4.6

$$B_d = \{x_a : a \in A_d \text{ there is a dance on } \hat{a}\}$$

It is easy to prove that the following monotonicity properties hold for B_d: (1) $B_{d-1} \subseteq B_d$; (2) $x_a \in B_{d-1}$ if and only if for all $i \notin dom(a)$, $x_a Q_i \in B_d$. In order to show that B_d is a basis for $Span(T_d)$ we need to define an order $\prec$ on polynomials in T_d. We will do it as in [46].

Definition 5.4.7 *Let x_a and x_b be two polynomials in T_d. then $x_a \prec x_b$ if and only if $deg(x_a) < deg(x_b)$, or if $deg(x_a) = deg(x_b)$, then for the largest pigeon i such that $a(i) \neq b(i)$, we have that $a(i) < b(i)$.*

Lemma 5.4.4 *B_d is a basis for $Span(T_d)$ for any $d \leq \frac{\log n}{2}$.*

Proof. Under the hypothesis that the degree d is less than $\frac{\log n}{2}$ we show: (1) that $|B_d| \leq |T_d|$ and (2) that any $x_a \in T_d$ can be expressed as a linear combination of elements of B_d, from which the Lemma follows. The first property follows because the minimal dance defines a 1-1 into mapping from B_d to T_d. More precisely, if $x_a \in B_d$ then we have a dance on $\hat{a}$ and since $d \leq \frac{\log n}{2}$, then by Lemma 5.4.3, there is dance on a and therefore by Lemma 5.4.1 there is a minimal dance on a that by Lemma 5.4.2 is a 1-1 mapping. Finally the observation in the previous Fact proves the first part. For the second part we work by induction on $\prec$. Assume that for all $x' \prec x_a$, $x' \in Span(B_d)$. We show that $x_a \in Span(B_d)$. If there is a dance on a then x_a is in B_d. Otherwise we show how to express x_a as a linear combination of the elements of B_d. Let P_t be the set of all possible correct first t steps of the dance on a. We prove that $x_a \in Span(B_d)$ iff $\sum_{b \in P_t} x_b \in Span(B_d)$ by induction on $t = 0, \ldots, n$. Since there is no dance on a, then $P_n = \emptyset$ and therefore the claim follows. The base of the induction $t = 0$ follows since $P_0 = a$. For the induction step observe that if $t \notin dom(a)$ then $P_t = P_{t-1}$ and so the claim follows by induction on t. Otherwise for any $b \in P_{t-1}$, x_b is of the form $x_{t,j}x_c$. We rewrite $x_{t,j}$ with respect to the relation Q_t, so that x_b can be rewritten as

$$(1) \quad x_c \perp x_c Q_t \perp \sum_{j' \neq j} x_c x_{t,j'}$$

equation 1 can be rewritten as:

$$x_c \perp x_c Q_t \perp \sum_{j'<j} x_c x_{t,j'} \perp \sum_{j'>j, j' \in Good(t,b)} x_c x_{t,j'} \perp \sum_{j'>j, j' \notin Good(t,b)} x_c x_{t,j'}$$

Each monomial in the last term is equal 0, therefore in $Span(B_d)$. The first three terms in the above sum are in $Span(B_b)$ by induction on $\prec$. The first by the base case of the definition of $\prec$. The second by the induction case of $\prec$ and by the monotonicity property of B_d. The third by (the second case of the definition) $\prec$. The fourth term corresponds exactly to all the possible correct first t steps of b. Therefore if we sum over all x_b for $b \in P_{t-1}$ we have that

$$\sum_{b \in P_t} x_b \in Span(B_d) \quad \text{iff} \quad \sum_{b \in P_{t-1}} x_b \in Span(B_d)$$

This concludes the proof of the Lemma. $\Box$

Theorem 5.4.1 *Any polynomial calculus refutation of $MPHP_n$ has degree not less than $\frac{\log n}{2}$.*

Proof. The proof is as in [46]. That is we prove by induction on the length of the proof that each polynomial derivable from the initial polynomials Q_i with at most degree d is a linear combinations of polynomials in $B_d \perp T_d$ (i.e a combination of the elements of B_d that are multiples of some axioms Q_i). Therefore since $1 \in T_d$ and it has a unique representation in each basis, we cannot derive the polynomial 1 with a proof of degree less than or equal to d.

Recall that we are considering refutation modulo the vector I. Therefore in the base case an axiom is always of the form Q_i for some $i \in [n]$, and the claim follows.

In the inductive step, if a line is inferred by the sum rule the result is immediate. For the case of product, say we have $\frac{x_a}{x_a x_{i,j}}$, with $|a| \leq d \perp 1$. We want to prove that $x_{i,j}x_a \in$

$Span(B_d \perp T_d)$. By induction, x_a can be written as a sum of elements in $B_d \perp T_d$ (i.e. multiples of Q_i). Therefore distributing $x_{i,j}$ along elements of this sum, we can write $x_a x_{i,j}$ as a sum of multiples of Q_i. By the monotonicity properties of B_d, it is easy to see that this is a sum of scalar multiples of Q_i and therefore in $Span(B_d \perp T_d)$. □

We have proved a $\Omega(\log n)$ degree lower bound for the polynomials (1)-(4) defined at the beginning of this section.

Consider the formula 3-$MPHP_n$ obtained from $MPHP_n$ the same way MGT_n is obtained from GT_n. Consider for all $i \in [n]$, $m+1$ new extension variables $y_{i,0}, \ldots, y_{i,m}$. Define 3-$MPHP$ by changing for all $i \in [n]$, the clauses $\bigvee_{i=1}^{m} x_{i,j}$ of $MPHP$ by the set of clauses:

$$\bar{y}_{i,0} \wedge \bigwedge_{j=1}^{m} (y_{i,j-1} \vee x_{i,j} \vee \bar{y}_{i,j}) \wedge y_{i,m}$$

First observe that, by exactly the same argument given in Theorem 5.2.2, we can show the following Theorem.

Theorem 5.4.2 *There are polynomial size unrestricted resolution refutations for the* 3-*$MPHP_n$.*

Consider the polynomial formulation of 3-$MPHP_n$, obtained simply applying the mapping tr to the clauses defining 3-$MPHP_n$. We can show a degree lower bound for PC proofs of this set of polynomials of the same order of that given for $MPHP$.

Theorem 5.4.3 *Any polynomial calculus refutation of* 3-$MPHP_n$ *require* $\Omega(\log_2 n)$.

Proof. We will prove that $MPHP_n$ is (1,3)-reducible to 3-$MPHP_n$ following the definitions of (d_1, d_2)-reductions from [24]. Define

$$y_{i,j} = 1 \perp \sum_{k>j}^{m} x_{i,k}$$

We prove that all the initial polynomials of 3-$MPHP_n$ (with y substituted as defined above) are derivable with a 3-degree polynomial calculus refutations from initial polynomials of $MPHP_n$.

Observe that $y_{i,0} = 1 \perp Q_i = 0$ and $y_{i,m} = 1$. So we can easily prove $y_{i,0} = 0$ and $1 \perp y_{i,m} = 0$. A generic initial polynomial of 3-$MPHP_n$ of the form

$$(1 \perp y_{i,j-1})(1 \perp x_{i,j})(y_{i,j}) \text{ for } 2 < j < n$$

is equivalent to

$$(\sum_{k>j-1}^{m} x_{i,k})(1 \perp x_{i,j})(\sum_{k>j}^{m} x_{i,k})$$

And this can be rewritten as

$$(1 \perp \sum_{k=1}^{j-1} x_{i,k})(1 \perp x_{i,j})(1 \perp \sum_{k>j}^{m})$$

By simple calculations (using the initial axioms of $MPHP_n$) this is equal to

$$(1 \perp Q_j) + \sum_{k=1,k\neq j}^{n} x_{i,k}x_{i,j} + (1 \perp x_{i,j})\sum_{k=1}^{j-1} x_{i,k} \sum_{k=j+1}^{m} x_{i,k}$$

Using the initial axioms of $MPHP_n$ it is easy to see that each of the three terms of the above polynomial is equal to 0. $\Box$

Given the size-width trade-off optimality result, it would be interesting to prove that some polynomial formulation of GT_n requires degree $\Omega(n)$ in Polynomial Calculus. This result would show that there is a tautology provable fast in Resolution, but requiring high degree in Polynomial Calculus. Recall the simulation Theorem of Clegg,Edmonds and Impagliazzo [29]

Theorem 5.4.4 *([29]) If a set of clauses $F = C_1, \dots, C_m$, with $|C_i| \leq k$, for all $i = 1, \dots, m$ and over n variables has a resolution proof with S lines, then the set of polynomials $tr(F)$ has PC refutation of degree at most $3\sqrt{n \log_e S} + k + 1$.*

The size-degree trade-off of the previous Theorem is optimal, since there is a trivial resolution proof of size $O(2^n)$ for the PHP_n^{n+1}, and Razborov in [69] showed that PHP_n^m require $\Omega(n)$ degree PC refutations for all m. The same result holds if we modify the PHP_n^m to a 3-PHP_n^m as we have done above for the $MPHP$. It follows that previous size-degree trade-off of Theorem 5.4.4 is optimal. But, what is more interesting, is whether this optimality can also be proven in the case S is polynomial in the size of the formula. Buss and Pitassi [25] showed polynomial size unrestricted resolution refutations for the PHP^m_n, when $m = 2^{\sqrt{n \log n}}$. This result, joint with Razborov degree lower bound for PHP^m_n imply that the trade-off of Theorem 5.4.4 cannot be better than a quasipolynomial factor. It is easy to see that the same result can be obtained using our degree lower bound for 3-$MPHP$ (Theorem 5.4.3).

Our conjecture is that the simulation of [29] is optimal for small resolution proofs. We think that some polynomial version of the formula GT_n should require $\Omega(n)$ degree in PC for some field.

5.5 Resolution lower bounds via degree lower bounds

The following Lemma shows that degree lower bounds imply width lower bounds as long as the initial polynomials of the PC proofs are a direct translation of the initial clauses of the resolution proofs.

Lemma 5.5.1 *Given a set of unsatisfiable clauses F and a resolution refutation of F, there is a polynomial calculus refutation of $tr(F)$ of degree less than or equal to $w(\vdash F)+1$.*

Proof. For a generic clause $A = A^+ \vee A^-$ where $A^+ = (a_{i_1} \vee \ldots \vee a_{i_k})$ and $A^- = (\bar{a}_{j_1} \vee \ldots \vee \bar{a}_{j_l})$, let $poly(A^+) = \prod_{\ell=1}^{i_k} (1 \perp a_\ell)$ and $poly(A^-) = \prod_{\ell=1}^{j_l} a_\ell$. Then $poly(A) = poly(A^+) \cdot poly(A^-)$. Observe that given two clauses A and B, it is easy to obtain a PC derivation of

$$Poly(A) = 0 \vdash Poly(A)Poly(B) = 0$$

with a degree equal to $w(A) + w(B)$.

We show that for each line A in the resolution proof we find a PC refutation of $Poly(A) = 0$ with degree $w(A) + 1$. If A is a initial clause the result follows by definition of tr. Now assume that at a resolution step we are in the following situation

$$\frac{A \vee x \quad \bar{x} \vee B}{D}$$

Assume that $A = A' \vee C$ and $B = B' \vee C$, i.e. C is the clause formed by the literals that belong to both A and B. Observe that $w(D) = w(A') + w(B') + w(C)$. By induction we have derived

$$Poly(A)(1 \perp x) = Poly(A')Poly(C)(1 \perp x) = 0 \quad \text{and} \quad Poly(B)x = Poly(B')Poly(C)x = 0$$

By the previous observation and using the inductive hypothesis we can obtain both the refutations

$$Poly(A')Poly(C)Poly(B')(1 \perp x) = 0 \quad \text{and} \quad Poly(B')Poly(C)Poly(A')x = 0$$

with degree equal to $w(A') + w(B') + w(C) + 1 = w(D) + 1$. An application of the sum rule gives $Poly(D) = 0$ and does not increment the degree of the polynomials involved. □

The previous lemma also shows that the degree lower bound obtained for 3-$MPHP_n$ cannot be improved. In fact [39] shows how to obtain a superpolynomial size resolution refutation of 3-$MPHP_n$ of width $O(\log n)$, and by the lemma there is also a polynomial calculus refutation of the direct translation of degree $O(\log n)$.

As a consequence of the previous lemma and the width-size trade-off [10] (see theorem 5.1.1), a linear (in the number of variables) degree lower bound in polynomial calculus can give us an exponential lower bound in resolution size.

Finally observe that the previous lemma is better (in the sense that it gives a smaller degree PC refutations) than the corresponding simulation lemma of [29] in the case we have constant width polynomial Resolution refutations of formulas having initial clauses of constant size. Moreover it implies that under the tr translation the width base algorithm of [10] cannot be better than the Grobner basis algorithm of [29].

It would be interesting to obtain the opposite direction of lemma 5.5.1. Buresh-Oppenheim and Pitassi [20] have a simulation of polynomial calculus by resolution when we start with binomial equations as initial polynomials. The simulation has the property that the width is twice the degree.

Chapter 6

Discussion and Open Problems

In this Chapter we summarize the results obtained in the thesis and we end with a list of open problems that we consider important and interesting.

6.1 Summary of Results

In the thesis we have investigated the complexity of proofs in several propositional proof systems. Our main motivation has been to contribute to the line of research started by Cook and Reckhow to obtain as much knowledge as possible about the complexity of different proof systems to show that there is no super proof system.

We also have been motivated by more applied questions concerning the automatic generation of Theorems. The results presented in the thesis were obtained in the papers [15, 14, 16]. We briefly summarize them by Chapters:

In Chapter 3 we have proved lower bounds for the number of lines for Substitution Frege proofs of some tautologies introduced by Urquhart in [80]. We use an approach based upon Kolmogorov Complexity to obtain this lower bound previously given by [80]. We extend the previous result to a linear lower bound in the case of the tree-like Substitution Frege system. We show that these lower bounds are optimal giving matching upper bounds. We extend the result also to another class of tautologies. Finally we show that tree-like substitution Frege systems polynomially simulate dag-like substitution Frege systems , and that Frege systems polynomially simulate tree-like Renaming Frege systems.

In Chapter 4 we have proved exponential separations between tree-like and dag-like version of both Resolution and Cutting Planes proof systems, improving the previous superpolynomial ones. The lower bounds are a consequence of extending the separation of the monotone NC hierarchy obtained in [66] to the case of real circuits. We also improve the superpolynomial separation of [38] between Davis-Putnam resolution and unrestricted resolution, to exponential.

In Chapter 5 we show that the relationship between size and width, recently obtained in [10], is optimal even for several restrictions of the resolution system. A consequence of our result is that there is a formula which provides an exponential separation between the tree-like resolution system and all the restrictions of resolution we consider. The separation from linear resolution is new and previously unknown. Moreover our result implies that a new algorithm proposed in [10] to seek for resolution proofs cannot be used to obtain automatizability of resolution. Finally, we give a degree lower bound for a formula encoding a modification of the Pigeon Hole Principle, which has polynomial size resolution refutations. The main contribution of this last result is to show that the technique of Razborov [69] to give degree lower bounds in the Polynomial Calculus, also holds for other formulas than the PHP_n^{n+1}.

6.2 Open Problems

One of the main problems in complexity of proofs is to obtain lower bounds for the size and for the number of lines better than the known ones (in [22]) for Frege systems. This is a very hard problem.

Problem 1 *Find superpolynomial lower bounds for the size and/or for number of lines in Frege systems.*

About Frege systems with substitution rules ($s\mathcal{F}$, $r\mathcal{F}$) we would like to point out the following problems. Recall from Chapter 3 that tree-like $s\mathcal{F}$ polynomially simulates dag-like $s\mathcal{F}$ and that the same holds for simple Frege systems $\mathcal{F}$ (see [12]) and for Extended Frege $e\mathcal{F}$ (see Chapter 3 or [31]). As observed in Chapter 3 a similar property is not expected to hold also for Renaming Frege $r\mathcal{F}$. Indeed, as already obseved, this would imply that $\mathcal{F}$ polynomially simulates $s\mathcal{F}$. We think that it would be interesting to investigate what is the particular feature of dag-like $r\mathcal{F}$ that would make it strictly more powerful that tree-like $r\mathcal{F}$. Why does the tree-like polynomial simulation hold for the full substitution rule, but should not hold for the $r\mathcal{F}$.

This kind of questions were also investigated by N. Arai in [3]. She studied further restrictions of the substitution rule, weaker than the renaming. She considered two rules:

- the *restricted renaming rule*, a renaming rule in which the substituting variables must not occur among the substituted variables, and
- the *permutation rule* which only allows to permute the variables in the formula on which we apply the substitution.

With J.L Esteban in [33] we note that a restricted renaming Frege system polynomially simulates the renaming Frege system $r\mathcal{F}$. But, a more important question is to know if the permutation rule is as powerful as the renaming rule. Indeed in the dag-like $r\mathcal{F}$ polynomial simulation of dag-like $s\mathcal{F}$, showed in [22], the main point is to allow a renaming which maps two different variables into the same one. It is not clear at all how to obtain this kind of renaming using a permutation. So

Problem 2 *Does a Frege system augmented with a permutation rule polynomially simulate the renaming Frege systems $r\mathcal{F}$?*

Another interesting problem is to study the complexity of proofs in a monotone Frege system. Monotone systems were considered by Pudlak and Buss in [65], Pudlak in [64], Clote and Setzer in [30]. There are several different definitions of monotone systems (see

[65, 64] for details); the easiest is that of define it as a sequent calculus for monotone sequents (i.e sequents made by formulas defined over the basis $\{\wedge, \vee\}$), where we drop the rules manipulating the $\neg$ connective. Since the PHP_n^{n+1} can be expressed as a monotone sequent, it is interesting to know what is the complexity of the PHP_n^{n+1} in a monotone sequent calculus (see [64, 4] for further details about the importance of this problem). In [4] we gave a partial answer to this problem showing that the PHP_n^{n+1} can be proved in a monotone sequent calculus with quasipolynomial size proofs. The following problem, posed by Pudlák in [64], therefore remains open.

Problem 3 *([64]) Does the PHP_n^{n+1} have polynomial size monotone proofs.*

Moving from Frege systems to Resolution systems, first of all we would like to point out that the separation between regular resolution and unrestricted resolution is still superpolynomial, by a work of Goerdt [40]. Therefore the following problem is open:

Problem 4 *Is there any class of unsatifiable CNF formulas requiring exponential size refutations in regular resolution, but provable with a polynomial size refutation in unrestricted resolution ?*

An important problem, as pointed out in several papers [6, 25, 23, 70, 54] is to give the exact size complexity of resolution proofs of the PHP_n^m, when $m > \frac{n^2}{\log_2 n}$. All the known techniques to give size lower bounds in unrestricted resolution fails on this formula. On the other hand it seems very unlikely that PHP_n^m, for $m > \frac{n^2}{\log_2 n}$ could have polynomial size resolution proofs. Only for $m = 2^{\sqrt{n \log_2 n}}$ Buss and Pitassi in [25] give polynomial size resolution refutation of PHP_n^m. The following problems probably will require the introduction of a new lower bound technique.

Problem 5 *Prove exponential (in the size of the formula) lower bounds in unrestricted resolution for the size of the smallest refutation of PHP_n^m, when $m = O(n^{O(1)})$.*

Problem 6 *([54]) Prove exponential (in n) lower bounds in unrestricted resolution for the size of the smallest refutation of PHP_n^m for all $m > n$.*

The Cutting Planes proof system is not super. Several results showed formulas requiring exponential size refutations in Cutting Planes [52, 17, 63]. Nevertheless, it is interesting to note that, contrary to the case of Resolution, where we have several technique (mostly originated from the Haken's one) to give proof size lower bound, at present no direct combinatorial proof of non-efficiency of Cutting Planes (even in the case of polynomially bounded coefficient) is known. According to Razborov it would be interesting to find such a result.

Problem 7 *([71]) Find a direct combinatorial technique to obtain exponential lower bounds in Cutting Planes.*

Finally we would like to end with a problem and a conjecture which arises from our paper [16]. Recall that the Polynomial Calculus is automatizable via the Grobner basis algorithm (see [29]). We would like to prove that in fact this algorithm is not so powerful for finding refutations, when compared with the resolution system via a standard translation of CNF into sets of polynomials. Therefore we would like to prove that there is some formula F over n variables "easy" to prove in unrestricted resolution, but such that on its standard translation the Grobner basis algorithm must run for an exponential time in n to find a refutation of F. This last result would be a consequence of showing a degree lower bound for the Polynomial Calculus refutation of F of the order $\Omega(\sqrt{n})$. Moreover this result would prove that the simulation of the Resolution systems by the Polynomial Calculus, given in [29] (see Chapter 5 for the exact formulation of this theorem), is optimal in the case we start we polynomial size resolution proofs. Therefore the following is an interesting problem:

Problem 8 *Find a formula F over n variables, such that: (1) F has polynomial size resolution refutations; and (2) the standard polynomial formulation of F over a field K which requires $\Omega(\sqrt{n})$ degree Polynomial Calculus refutations.*

We conjecture that some polynomial formulation of our formula GT_n or MGT_n will require $\Omega(n)$ degree Polynomial Calculus refutations.

Bibliography

[1] M. Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 346–355, 1988.

[2] N. Alon and R. Boppana. The complexity of the pigeonhole principle. *Combinatorica*, 14:417–433, 1994.

[3] N. Arai. Tractability of cut-free gentzen type propositional calculus with permutation inference. *Theoretical Computer Science*, 170:129–144, 1996.

[4] A. Atserias, N. Galesi, and R. Gavalda. Monotone proofs of the pigeon hole principle. *Manuscript Submitted*, 1999. Available as TR n. in the Dept LSI at the web page http://www-lsi.upc.es/dept/techreps/1999.html.

[5] R.A. Baeza-Yates, R. Gavaldá, and G. Navarro. Bounding the expected length of longest common subsequences and forest. In *Third South American Workshop on String Processing (WSP96)*, 1996.

[6] P. Beame and T. Pitassi. Simplified an improved resolution lower bounds. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 274–282, 1996.

[7] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 200–220, Victoria, British Columbia, Canada, 4–6 May 1992.

[8] S. Bellantoni, T. Pitassi, and A. Urquhart. Approximations and small depth frege proofs. *SIAM Journal on Computing*, pages 1162–1179, 1992.

[9] E. Ben-Sasson and R. Impagliazzo. Random cnf's are hard for polynomial calculus. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 415–421, 1999.

[10] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 517–526, 1999. To appear in Journal of the ACM. Also available as ECCC TR n. TR99-022 at the web page http://www.eccc.uni-trier.de/eccc-local/Lists/TR-1999.html.

[11] M.L. Bonet. *The Lengths of Propositional Proofs and the Deduction Rule.* PhD thesis, University of California, Berkeley, 1991.

[12] M.L. Bonet and S. R. Buss. The deduction rule and linear and near-linear proof simulations. *Journal of Symbolic Logic*, 58(2):688–709, 1993.

[13] M.L. Bonet, S. R. Buss, and T. Pitassi. Are there hard examples for frege proof systems? In P. CLote and F. Remmel, editors, *Feasible Mathematics II*, pages 30–56. Birkhauser, 1995.

[14] M.L. Bonet, J.L. Esteban, N. Galesi, and J. Johannsen. Exponential separations between restricted resolution and cutting planes proof systems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 638–647, 1998. Submitted for publication to a journal.

[15] M.L. Bonet and N. Galesi. Linear lower bounds and simulations in frege systems with substitution. In M. Nielsen and W. Thomas, editors, *Selected Papers of 11-th Computer Science Logic-Lecture Notes in Computer Science, 1414*, pages 115–128, 1998.

[16] M.L. Bonet and N. Galesi. A study of proof search algorithms for resolution and polynomial calculus. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 422–431, 1999.

[17] M.L. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proof with small coefficients. *Journal of Symbolic Logic*, 62(3):708–727, 1997. Preliminary Version appeared in the *ACM Symposium on the Theory of Computing*, STOC'95. Las Vegas, Nevada, USA. 1995.

[18] M.L. Bonet, T. Pitassi, and R. Raz. No feasible interpolation for tc^0- frege proofs. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 254–265, 1997.

[19] Boppana and Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Ed. Jan van Leeuwen, Volume A: Algorithms and Complexity)*, volume 1. Elsevier and MIT Press, 1990.

[20] J. Buresh-Oppenheim and T. Pitassi. Some remarks on the polynomial calculus and resolution proofs. *Manuscript in Preparation*, 1999.

[21] S. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52:916–927, 1987.

[22] S. Buss. Some remarks on lengths of propositional proofs. *Archive for Mathematical Logic*, 34:377–394, 1995.

[23] S. Buss. Lectures on proof theory. Technical report, McGill University, 1996.

[24] S. Buss, D. Grigoriev, R. Impagliazzo, and T. Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 547–556, 1999.

[25] S. Buss and T. Pitassi. Resolution and the weak pigeonhole principle. In M. Nielsen and W. Thomas, editors, *Selected Papers of 11-th Computer Science Logic-Lecture Notes in Computer Science, 1414*, pages 149–156, 1998.

[26] S. Buss and G. Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–317, 1988.

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

[28] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the Association for the Computing Machinery*, 35(4):759–768, October 1988.

[29] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, Philadelphia, Pennsylvania, 22–24 May 1996.

[30] P. Clote and A. Setzer. On *PHP*, *st*-connectivity and odd charged graphs. In P. Beame and S. R. Buss, editors, *Proof Complexity and Feasible Arithmetics*, pages 93–117. AMS DIMACS Series, 1988.

[31] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.

[32] W. Cook, C.R. Coullard, and G. Turán. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

[33] J.L. Esteban and N. Galesi. A note on renamings in frege systems. *Manuscript*, 1998.

[34] X. Fu. Lower bounds on sizes of cutting planes proofs for modular coloring principles. In P. Beame and S. R. Buss, editors, *Proof Complexity and Feasible Arithmetics*, pages 135–148. AMS DIMACS Series, 1988.

[35] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[36] Z. Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4(1):23–46, February 1977.

[37] J. H. Gallier. *Logic for computer science.* Harper & Row Publishers, New York, 1986.

[38] A. Goerdt. Davis-Putnam resolution versus unrestricted resolution. *Annals of Mathematics and Artificial Intelligence*, 6:167–184, 1992.

[39] A. Goerdt. Unrestricted resolution versus N-resolution. *Theoretical Computer Science*, 93:159–167, 1992.

[40] A. Goerdt. Regular resolution versus unrestricted resolution. *SIAM Journal on Computing*, 22:661–683, 1993.

[41] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[42] A. Haken and S. Cook. An exponential lower bound for the size of monotone real circuits. *Journal of Computer and Systems Science*, 1998. To appear.

[43] J. Hastad. *Computational Limitations for Small Depth Circuits.* The MIT press, 1986.

[44] J. R. Hindley and D. Meredith. Principal type schemes and condensed detachment. *Journal of Symbolic Logic*, 55:90–105, 1990.

[45] R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings of the Symposium on Logic in Computer Science (LICS) IEEE Symposium on Foundations of Computer Science*, pages 220–228, 1994.

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

[47] K. Iwana and T. Pitassi. Exponential lower bounds for the tree-like hajós calculus. *Manuscript*, 1997.

[48] J. Johannsen. Lower bounds for monotone real circuit depth and formula size and tree-like cutting planes. *Information Processing Letters*, 67:37–41, 1998.

[49] S. Jukna. Combinatorics of monotone computations. *Combinatorica*, 1:65–85, 1999.

[50] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Computing*, 3:255–265, 1990. Preliminary version in *Proc. 20th ACM Symposium on Theory of Computing*, 1988.

[51] J. Krajìček. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, 52(1):73–86, 1994.

[52] J. Krajìček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62:457–486, 1997.

[53] J. Krajìček. Interpolation by a game. *Mathematical Logic Quarterly*, 44:450–458, 1998.

[54] J. Krajìček. On the weak pigeonhole principle. *Manuscript submitted*, 1999. Available at the author web page http://www.math.cas.cz/ krajicek/.

[55] J. Krajìček and P. Pudlàk. Propositional proof systems, the consistency of first order theories and the complexity of computation. *Journal of Symbolic Logic*, 54:1063–1079, 1989.

[56] J. Krajìček, P. Pudlàk, and A. Woods. Exponential lower bounds to the size of bounded depth frege proofs of the pigeon hole principle. *Computational Complexity*, 1994.

[57] B. Krishnamurthy. Short proofs for tricky formulas. *Acta informatica*, 22:253–275, 1985.

[58] M.Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Application.* Springer Verlag, 1993.

[59] V.P. Orevkov. On lower bounds on the lengths of proofs in propositional logic. *in Proc. of All Union Conf. Metody Matem. v Problemach iskusstvennogo intellekta i sistematicheskoje programmirovanie*, 1:142–144, 1980.

[60] V.P. Orevkov. Reconstruction of a proof from its scheme. *Soviet Mathematics Doklady*, 35:326–329, 1987.

[61] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

[62] A. N. Prior. *Formal Logic*, volume Second Edition. Oxford, 19460.

[63] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62:981–998, 1997.

[64] P. Pudlák. On the complexity of propositional calculus. In *Sets and Proofs*, volume 62, pages 197–218. Cambridge University Press, 1999. Also invited paper from *Logic Colloquium 1997*.

[65] P. Pudlak and S.R. Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In *8th Workshop on Computer Science Logic, Kazimierz, Poland*, pages 151–162. Springer - Lecture Notes in Computer Science, 1995.

[66] R. Raz and P. McKenzie. Separation of the monotone *NC* hierarchy. *Combinatorica*, 19(3):403–435, 1999. Preliminary version in Proc. of 38th Symposium on Foundations of Computer Science, pages 234–243, 1997.

[67] A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Dokl. Ak. Nauk. SSSR*, 281:798–801, 1985. In Russian. english Translation in *Sov. Math. Dokl.* Vol 31, pp. 354-357.

[68] A. Razborov. Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic. *Izvestiya of the R.A.N. Dokl. Ak. Nauk. SSSR*, 59(1):210–224, 1995.

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

[70] A. Razborov, A. Wigderson, and A. Yao. Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 739–748, El Paso, Texas, 4–6 May 1997.

[71] A. A. Razborov. Lower bounds for propositional proofs and independence results in bounded arithmetic. *Lecture Notes in Computer Science*, 969:105–??, 1995.

[72] J. A. Robinson. A machine oriented logic based on the resolution principle. *Journal of the Association for the Computing Machinery*, 12:627–631, 1965.

[73] A. Rosembloom. Monotone real circuits are more powerful than monotone boolean circuits. *Information Processing Letters*, 61:161–164, 1997.

[74] U. Schoning. *Logic for Computer Scientists*. Birkhauser, 1989.

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

[76] G.S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 2:115–125, 1970.

[77] A. Urquhart. Hard examples for resolution. *Journal of the Association for the Computing Machinery*, 34:209–219, 1987.

[78] A. Urquhart. The relative complexity of resolution and cut-free gentzen systems. *Annals of Mathematics and Artificial intelligence*, 6:157–168, 1992.

[79] A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1:425–467, 1995.

[80] A. Urquhart. The number of lines in frege proof with substitution. *Archive for Mathematical Logic*, 1(1):15–19, 1997.

[81] Ingo Wegener. *The Complexity of Boolean Functions.* B.G. Teubner, Stuttgart, 1 edition, 1987.