Space in Weak Propositional Proof Systems

Ilario Bonacina

Space in Weak Propositional Proof Systems

Springer

Ilario Bonacina
Departament de Ciències de la Computació
Universitat Politècnica de Catalunya
Barcelona
Spain

ISBN 978-3-319-73452-1 ISBN 978-3-319-73453-8 (eBook)
https://doi.org/10.1007/978-3-319-73453-8

Library of Congress Control Number: 2017963852

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

to my family

Preface

In this book you, the reader, are going to see some results on the *space* complexity of some propositional proof systems. This book is a revised version of my PhD thesis[1] and indeed it is not intended to be a survey of *all* the results known on the space complexity of propositional proof systems. It will rather be a long walk touching some topics in proof complexity, mostly about space of course, but not exclusively. Hopefully this could be used too as a rather reader-friendly exposition of some game theoretic methods used in proof complexity. This is indeed an underground theme connecting most of the results we show. Of course there will be some survey(-ish) parts but mainly the focus will be on the new game theoretic techniques and their application to the analysis of the space complexity of propositional proof systems. That is the results arising from my PhD thesis [Bon15] and some earlier works [BG13, BGT14, BT15, BG15, BBG$^+$17, BT16a, BT16b, BGT16, Bon16].

This is a work about proof complexity, so let's start by introducing it informally. *Proof complexity* is a research area that studies the concept of complexity from the point of view of logic. In particular, in proof complexity we are interested in questions such as: "*how difficult is it to prove a theorem?*" Or, more precisely, given a formal system, we are interested in measuring the complexity of a theorem, that is answering questions such as "*what is the shortest proof of the theorem in a given formal system?*" This mirrors questions in computational complexity about, for example, the number of steps that a Turing machine needs to compute a given function f; or the size of circuits needed to compute f.[2]

In this book we investigate the space complexity of propositional proof systems, so what is the *space* of a proof? We could state this question pictorially as "*what is the smallest blackboard a teacher needs to present the proof of a theorem to a class of students?*"[3] As before, this notion is analogous to the space complexity

[1] This revised version is due to the fact that my thesis was awarded "*Best Italian PhD Thesis in Theoretical Computer Science*" for the year 2016 by the Italian chapter of the European Association for Theoretical Computer Science (EATCS).

[2] On the other hand, we could also measure the complexity of a theorem as the strength of a theory needed to prove the theorem. This also has a counterpart in computational complexity, it is linked with questions about the smallest complexity class to which a given function belongs.

[3] We suppose here that the students can understand just proofs written on the blackboard in some given formal system and they do not have any additional memory except the minimal one to

in the context of uniform computations, measured, for example, as the size of a working-tape needed by a Turing machine to compute a given function.

Propositional proof complexity, that is the complexity of propositional proofs, plays a role in the context of feasible proofs as important as the role of Boolean circuits in the context of efficient computations. Although the original motivations to study the complexity of propositional proofs came from proof-theoretical questions about first-order theories, it turns out that, essentially, the complexity of propositional proofs deals with the following question: "*what can be proved by a prover with bounded computational abilities?*" For example, if its computational abilities are limited to small circuits from some circuit class. Hence, propositional proof complexity mirrors non-uniform computational complexity and indeed there is a very productive cross-fertilization of techniques between the two fields. Our understanding of propositional proof systems is, unfortunately, similar to the general situation in complexity theory. In both fields we can prove lower bounds in very special cases and indeed there are several major open problems that are very basic, way more basic than the well-known question $\mathsf{P} \overset{?}{=} \mathsf{NP}$. The situation is similar in the sense that we can prove super-polynomial lower bounds on the length of proofs only for restricted proof systems. Indeed, by a result of [CR79], proving super-polynomial lower bounds on the length of proofs for *every* propositional proof system is equivalent to showing that $\mathsf{NP} \neq \mathsf{coNP}$, which in turn is one of the open and very important problems in computational complexity. Propositional proof complexity is important also from the practical point of view. The implementations of state-of-the-art SAT algorithms ultimately rely on rather simple propositional proof systems. Hence the study of those systems helps in clarifying the limitations of such algorithms that are essential in various aspects of computer science, cf. [Nor15].

We will focus on the space complexity of two particular proof systems: *resolution*, a well studied proof system that is at the core of state-of-the-art SAT-solvers; and *polynomial calculus*, a proof system that uses polynomials to refute propositional formulas that are contradictions. We will show some generic combinatorial techniques to prove space lower bounds in both those systems and then we will apply those techniques to show concrete space lower bounds for refutations of several particular (unsatisfiable) propositional formulas. Since the very first exponential size lower bound for resolution size in [Hak85], game theoretic methods and combinatorial characterisations of hardness measures have a long history in proof complexity. This book could be seen as the latest contribution to this topic.

For resolution the new techniques we introduce allowed for the first time to obtain—in a quite easy way actually—lower bounds for the space of proofs when the space is measured as the total number of variables to be kept in memory (*total space*). For polynomial calculus the techniques we introduce—which is more involved than those for resolution—allow us to address space lower bounds when the space takes into account the number of distinct monomials to be kept in memory (*monomial space*). Notably those techniques allow us to prove, among other results, that almost

understand the content of the blackboard. Moreover the teacher has to write with fonts of a fixed size.

all k-CNF formulas are hard with respect to total space in resolution and monomial space in polynomial calculus. That is we prove asymptotically optimal lower bounds for the monomial space (and total space in resolution) for random k-CNF formulas in n variables and a linear number of clauses. This was an open problem mentioned for the first time in [BS01, ABRW02] and since then reported many times in the literature.

Book Structure After an introduction to propositional proof complexity (Chap. 1), this work consists of 3 parts. Each chapter ends with a section containing open questions and a **History** section collecting some facts about the main theorems of the chapter and how they fit in the previous literature.

In **Part I** there are two chapters on resolution: one containing results already known in the literature before this work (Chap. 2) and one just focused on space in resolution (Chap. 3). More precisely on the combinatorial techniques to prove total space lower bounds. Then we move to polynomial calculus and its space complexity (Chap. 4). The focus will be now on the combinatorial technique to prove monomial space lower bounds.

In **Part II** we collect the main applications of the techniques we built previously. First there is a short chapter about the proof complexity and space complexity of the pigeon principles (PHP_n^m and its variations), cf. Chap. 5. Then there is an interlude on some new type of games, the *cover games*, defined on bipartite graphs (Chap. 6). This chapter is essentially independent from the rest of the book and it collects some results on graph theory. The motivation behind this chapter though is that the results in it will be needed in Chap. 7 to prove the space lower bound for random k-CNF formulas and other graph-based propositional formulas.

In the last part, **Part III**, we analyse the size of resolution proofs in connection with the *Strong Exponential Time Hypothesis* (SETH) in complexity theory. More precisely we prove strong size lower bounds for a restricted version of resolution we call δ-regular resolution. Although not directly related to space, the results we show here rely on some combinatorial characterisations and games analogous to the one used to prove space lower bounds.

Acknowledgments First of all, I want to thank my advisor Nicola Galesi at Sapienza University of Rome. He cultivated my interest in proof complexity supporting the good ideas and shooting down the (many) buggy ones. This book simply wouldn't have existed without him.

I am truly grateful to the many people from the proof complexity community that I had the privilege to meet. My view of proof complexity and theoretical computer science was shaped by all of the discussions we had. In particular I want to thank (the order is not relevant here): Jakob Nordström, Massimo Lauria, Mladen Mikša, Marc Vinyals, Susanna Figueiredo de Rezende, Navid Talebanfard, Tony Huynh, Paul Wollan, Pavel Pudlák, Jan Krajíček, Neil Thapen, Olaf Beyersdorff, Rahul Santhanam, Albert Atserias, Jacobo Torán, and Yuval Filmus.

Stockholm, *Ilario Bonacina*
May 31, 2017

Contents

1 **Introduction** 1

1.1 Propositional Proof Systems 1

1.1.1 Frege Systems 3

1.1.2 Algebraic Proof Systems 5

1.1.3 Geometric Proof Systems 8

1.1.4 Connection with SAT-Solvers 8

1.2 Space of Proofs 10

1.3 Summary of Results 11

Part I General Results and Techniques

2 **Resolution** 15

2.1 Subsystems of Resolution 15

2.2 Size 16

2.3 Width and Asymmetric Width 20

2.3.1 Combinatorial Characterizations 23

2.3.2 The “Equivalence” of Width and Asymmetric Width 26

2.4 Open Problems 28

3 **Space in Resolution** 29

3.1 Semantic Resolution 30

3.2 Clause Space 31

3.2.1 Semantic Clause Space 33

3.3 Total Space 34

3.3.1 Semantic Total Space 36

3.4 Space and Depth 38

3.5 Open Problems 39

4 **Space in Polynomial Calculus** 41

4.1 From CNF Formulas to Polynomials 42

4.1.1 Complete Tree Formulas CT_n 44

4.2 Size and Degree 44
4.3 Space 45
4.4 Semantic Polynomial Calculus 47
4.5 Monomial Space Lower Bounds 48
4.5.1 CT_n Requires Large Monomial Space 51
4.5.2 Proofs of Theorem 4.2 and Theorem 4.3 52
4.6 Open Problems 56

Part II Applications

5 Pigeonhole Principles 61
5.1 The (Standard) Pigeonhole Principles 61
5.2 The Bit-Pigeonhole Principle 65
5.3 The XOR Pigeonhole Principle 67
5.4 Open Problems 69

6 Interlude: Cover Games 71
6.1 $\mathcal{C}$-Matchings 71
6.2 Some Hall's Theorems 72
6.2.1 A Hall's Theorem for VW-Matchings 73
6.3 Cover Games 75
6.4 Random Bipartite Graphs 77
6.5 Winning Strategies for Cover Games 79
6.5.1 A Winning Strategy for the Game on V-Matchings 79
6.5.2 A Winning Strategy for the Game on VW-Matchings 83

7 Some Graph-Based Formulas 89
7.1 From Cover Games to Space Lower Bounds 89
7.2 Random *k*-CNF Formulas 92
7.3 Pigeonhole Principles over Graphs 95
7.4 Tseitin Formulas 97
7.5 Open Problems 100

Part III A Postlude

8 Strong Size Lower Bounds for (a Subsystem of) Resolution 105
8.1 SETH and Proof Complexity 105
8.2 Game Characterizations of Width and Size 108
8.3 A Strong Width Lower Bound 113
8.4 Open Problems 117

References 119

Index 129

List of Figures

2.1 An example of minimal-size resolution refutation of F 17
2.2 A canonical decision tree 18
2.3 Size upper bound via canonical decision trees 19

4.1 Simulation of the rule $\frac{C\vee x, \quad D\vee\neg x}{C\vee D}$ in polynomial calculus 43
4.2 A locality lemma 55

6.1 D_4 73
6.2 List of forbidden subgraphs for G 74
6.3 Component removal for V-matchings 80
6.4 Covering a vertex in L via V-matchings 81
6.5 Covering a vertex in U via V-matchings 82
6.6 Component removal for VW-matchings 84
6.7 Covering a vertex in L via VW-matchings 85
6.8 Covering a vertex in U via VW-matchings 86

7.1 From $\mathcal{C}$-matchings to flippable products: a minimal example 91
7.2 From $\mathcal{C}$-matchings to flippable products: inductive step 91

Notation

In this section we give the notation that shall be standard throughout this book.

Sets

We use the standard set-theoretic notations.

- $|S|$ is the size of the set S
- $[n]$ is the set of natural numbers $\{1,\dots,n\}$
- $\emptyset$ is the empty set
- $A \cup B = \{x \,:\, x \in A \text{ or } x \in B\}$
- $A \cap B = \{x \,:\, x \in A \text{ and } x \in B\}$
- $A \dot{\cup} B = A \cup B$ in the case $A \cap B = \emptyset$
- $A \setminus B = \{x \,:\, x \in A \text{ and } x \notin B\}$
- $A \subseteq B$ if every element of A is also an element of B
- (a,b) is an ordered pair of elements
- $A \times B = \{(x,y) \,:\, x \in A \text{ and } y \in B\}$
- $\binom{S}{2}$ is the set of subsets of the set S of size 2

Arithmetic

As customary, $\mathbb{N}$ is the set of all natural numbers, $\mathbb{R}$ is the set of real numbers, $\mathbb{F}$ is a generic field and $\mathbb{F}_p$ is a finite Galois field with p elements. Given a field $\mathbb{F}$, $\mathrm{char}(\mathbb{F})$ is the smallest integer a such that for every element x of $\mathbb{F}$, $\underbrace{x + \cdots + x}_{a} = 0$.

If not stated otherwise e will be the base of natural logarithms, $\mathrm{e} = 2.718\ldots$ We denote as $\ln(\cdot)$ the natural logarithm and with $\log(\cdot)$ the logarithm over base 2. Given a real number x, $\lfloor x \rfloor$ is the largest integer smaller or equal to x. The binomial coefficient $\binom{m}{n}$ is $\frac{m!}{n!(m-n)!}$. We use sometimes the inequality $\binom{m}{n} \leqslant \left(\frac{\mathrm{e}m}{n}\right)^n$.

Asymptotic notations. Given two functions f, g from $\mathbb{N}$ to $\mathbb{N}$ we say that $f = O(g)$ if there are some absolute constants c_1, c_2 such that for every $n \in \mathbb{N}$, $f(n) \leqslant c_1 g(n) + c_2$. We say that $f = \Omega(g)$ if $g = O(f)$ and $f = \Theta(g)$ if both $f = \Omega(g)$ and $f = O(g)$. We say that $f = \widetilde{O}(g)$ if there exists a $k \in \mathbb{N}$ such that $f = O\left(g \log^k g\right)$. We say that $f = o(g)$ if $\frac{f(n)}{g(n)} \to 0$ as $n \to \infty$. We say that $f = \omega(g)$ if $g = o(f)$. We say that f is *super-polynomial* in n if $f = n^{\omega(1)}$.

Logic

Propositional formulas. A Boolean variable x and its negation $\neg x$ are sometimes denoted respectively as x^1 and x^0. A *literal* ℓ is a Boolean variable or the negation of a Boolean variable. A disjunction of literals $\bigvee_{i \in I} \ell_i$ is a *clause*. Its size $|C|$ is the number of distinct literals in C. If $|C| \leqslant k$ we say that C is a *k-clause*. A conjunction of clauses $\{C_i : i \in [m]\}$ is a formula in *Conjunctive Normal Form* (CNF formula) and it is denoted also as $C_1 \wedge \cdots \wedge C_m$. If all the clauses are k-clauses then we say that the formula is a *k-CNF formula*. Given a CNF formula F, the set of Boolean variables mentioned in F is $\mathrm{vars}(F)$. The number of clauses mentioned in the CNF formula F is $|F|$.

We often consider families of formulas $(F_n)_{n \in \mathbb{N}}$ where usually $n = |\mathrm{vars}(F_n)|$ or n is polynomially related to $|\mathrm{vars}(F_n)|$. With a slight abuse of notation a family of formulas $(F_n)_{n \in \mathbb{N}}$ is denoted simply as F_n.

Boolean assignments. Given a set of variables X, a *Boolean assignment* over X is a map $\alpha : X \to \{0, 1, \star\}$, where X is a set of variables. The *domain* of α is $\mathrm{dom}(\alpha) = \alpha^{-1}(\{0,1\})$. We say that α is *assigning* a value to x if and only if $x \in \mathrm{dom}(\alpha)$. With λ we denote the unique Boolean assignment with empty domain.

Given a Boolean assignments α over X and α' over X', their *union* $\alpha \cup \alpha'$ is the following Boolean assignment over $X \cup X'$

$$\alpha \cup \alpha'(x) = \begin{cases} \alpha(x) & \text{if } x \in \mathrm{dom}(\alpha) \setminus \mathrm{dom}(\alpha') \\ \alpha'(x) & \text{if } x \in \mathrm{dom}(\alpha') \setminus \mathrm{dom}(\alpha) \\ \alpha(x) & \text{if } x \in \mathrm{dom}(\alpha) \cap \mathrm{dom}(\alpha') \text{ and } \alpha(x) = \alpha'(x) \\ \star & \text{otherwise} . \end{cases} \tag{0.1}$$

Given a Boolean assignment α over X and $Y \subseteq X$, the *restriction* $\alpha\restriction_Y$ is the Boolean assignment

$$\alpha\restriction_Y(x) = \begin{cases} \alpha(x) & \text{if } x \in Y \\ \star & \text{otherwise} . \end{cases} \tag{0.2}$$

Given two Boolean assignments α and α', we say that $\alpha \subseteq \alpha'$, if $\alpha'\restriction_{\mathrm{dom}(\alpha)} = \alpha$.

Evaluation of CNF formulas. Given a CNF formula F and a Boolean assignment α over $\mathrm{vars}(F)$, we can apply α to F obtaining a new CNF formula $F\restriction_\alpha$ in this way: for each variable $x \in \mathrm{dom}(\alpha)$ substitute x in F with the value $\alpha(x)$, otherwise leave

x untouched. Then simplify the resulting formula with the usual rules: $0 \vee C \equiv C$, $1 \vee C \equiv 1$, $0 \wedge C \equiv 0$, $1 \wedge C \equiv C$. We say that α *satisfies* F if $F\restriction_{\alpha} = 1$. We denote this as $\alpha \vDash F$. Similarly, for a family A of Boolean assignments we write $A \vDash F$ if for each $\alpha \in A$, $\alpha \vDash F$.

Algebra

Given a field $\mathbb{F}$ and a set of variables X, the ring $\mathbb{F}[X]$ is the ring of polynomials in the variables X with coefficients in $\mathbb{F}$. An *ideal* I in $\mathbb{F}[X]$ is any subset of $\mathbb{F}[X]$ closed under addition, $p,q \in I$ implies that $p+q \in I$, and closed under multiplication with elements of $\mathbb{F}[X]$, $p \in I$ and $q \in \mathbb{F}[X]$ implies that $pq \in I$. Given a set of polynomials P, $\langle P \rangle$ is the ideal generated by P in $\mathbb{F}[X]$. Given two ideals I,J in $\mathbb{F}[X]$, $I + J = \{a+b \ : \ a \in I \text{ and } b \in J\}$.

Evaluations of polynomials. Given a polynomial p in $\mathbb{F}[X]$ and a Boolean assignment α we define the *restriction* $p\restriction_{\alpha}$, as follows: for each variable $x \in \mathrm{dom}(\alpha)$ substitute x in p with the value $\alpha(x)$, or otherwise leave the variable untouched. Then simplify the result with the usual simplification rules including: $0 \cdot m \equiv 0$, $1 \cdot m \equiv m$ and $m - m \equiv 0$ where m is any term in p, that is any monomial with a coefficient from $\mathbb{F}$ in front of it.

Graphs

A *graph* G is a pair (V,E) where V is a set and $E \subseteq \binom{V}{2}$. The elements of V are called *vertices* of G and the elements of E are called *edges* of G. Given a vertex $v \in G$, the *neighbor* of v in G is $N_G(v) = \{w \in V \ : \ \{v,w\} \in E\}$. The size of $N_G(v)$ is the *degree* of v in G.

A graph G is a *bipartite graph* if there exists two disjoint sets L and U such that $V = L \dot{\cup} U$ and $E \subseteq \{\{v,w\} \ : \ v \in L \text{ and } w \in U\}$. The pair (L,U) is a *bipartition* of the bipartite graph G.

A *matching* in G is a set $M \subseteq E$ such that all the edges in M are pair-wise disjoint. A matching *covers* $S \subseteq V$ if for each $v \in S$ there exists $e \in M$ such that $v \in e$.

A standard result about matchings in bipartite graphs is **Hall's theorem**: given any bipartite graph G with bipartition (L,U), the following are equivalent

1. G has a matching covering L;
2. for every subset $S \subseteq L$, $|N_G(S)| \geqslant |S|$.

Bipartite expansion. Let $r \in \mathbb{N}$ and $c \in \mathbb{R}$. A bipartite graph G with bipartition (L,U) is a *(r,c)-bipartite expander* if and only if

$$\forall A \subseteq L(G), |A| \leqslant r \rightarrow |N_G(A)| \geqslant c\,|A| \ . \tag{0.3}$$