UNIFIED LOWER BOUNDS FOR MONOTONE COMPUTATION

by

Robert Robere

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Department of Computer Science
University of Toronto

Abstract

Unified Lower Bounds for Monotone Computation

Robert Robere
Doctor of Philosophy
Department of Computer Science
University of Toronto
2018

Razborov [55] introduced an elegant rank-based complexity measure for proving lower bounds on the monotone formula complexity of boolean functions. Later work by Gál [25] showed that Razborov's rank measure also provides lower bounds on *monotone switching networks*, which measure space complexity, and *monotone span programs*, which are an elegant complexity measure that have connections to cryptography. Despite this, the only lower bounds known for the rank measure are the original $n^{\Omega(\log n)}$ bounds for function computable in NP due to Razborov [55].

In this thesis, we give a new analysis of the rank measure; in particular, connecting it to the *Nullstellensatz degree* from propositional proof complexity. This ultimately allows us to "lift" Nullstellensatz degree lower bounds to rank measure lower bounds in a generic way, and using this lifting theorem we are able to prove a variety of new lower bounds in monotone complexity theory, as well as obtaining new proofs of many old lower bounds.

To prove our results we introduce several technical tools which we hope will have other applications. In particular, we introduce a new algebraic complexity measure called the *algebraic gap complexity* which is crucial in our reductions, as well as a generic way to transform multilinear polynomials into matrices such that the rank of the matrix can be directly calculated from the monomials in the polynomial (this generalizes an earlier result of Sherstov [61]).

Acknowledgements

Academically, I would not be where I am today without the help of many wonderful supervisors over the years. First and foremost, I will be forever thankful to both Toni and Stephen for their excellent supervision in my graduate work. It should go without saying that they are both first-rate scientists: excellent taste in problems, full of good advice, and seemingly endless knowledge about mathematics. But what strikes me most about both of them is their amazing attitudes: they care about their research because they like to think about cool things and then share those things with other people. What more could you ask for in an advisor? If I have picked up anything from them, I hope it is this.

Next down the list (academically), I would not even be in this program without the help of my undergraduate advisors, Todd Wareham and Antonina Kolokolova. It's crazy to think that all of this ultimately boils down to a chat with Todd ("So, what are you working on right now?") in the hallway one summer afternoon during a programming job at Memorial. On the other hand, Antonina is responsible for so much by now that I'm not even sure where to begin — thanks for the research support, for introducing me to the community, and for being a great friend.

I would also like to thank a number of folks in the community for some very interesting conversations over the years — Yuval Filmus, for good chats and for being a good pal since my Masters; Ben Rossman, for lots of discussions about circuit complexity and for serving on my final exam committee; Vijay Ganesh, for so, so many interesting chats about SAT solving; and Jakob Nordström, for inviting me to visit Sweden and many subsequent discussions about complexity theory in general and proof complexity in particular.

Life isn't all academics, of course, and for keeping me sane I have to thank all of my friends in Toronto. In no particular order: the lab (Lalla, Noah, Mika, Atiyeh, Spooner, Akis, David(s), Dustin, . . .), the climbing crew (Adam, Mike Pohlod, Sam, Alex, Kwok, Evan, JR, Danny, Rodrigo, Kirby, and Hans), and my fellow Newfie expatriates (Ryan, Evan (again), Nick, Danny Dillabough, Mike Fardy, Mitch, Ciarán, Andy, Wyatt, Big Dan, and Noah (again)). I also wouldn't have made it through this without Kate: thank you for the support, for putting up with me ("You look like you're thinking about math"), but most of all, just for being yourself.

Finally, I would like to thank my parents for their endless support through this (admittedly strange) path that I've chosen.

Thanks again. Without ya'll life would be much less colourful.

Contents

Chapter 1

Introduction

1.1 Complexity Theory

This is a thesis in *computational complexity theory*, which is the area of theoretical computer science that seeks to classify computational tasks by the resources — such as computation time or the amount of memory — required to solve them. Formally, this is done by studying *complexity classes*, which are collections of computational tasks that can be solved by an algorithm using a bounded amount of a computational resource. For example, the class P, (which stands for *polynomial time*), contains all tasks that can be solved by an algorithm using a polynomially-bounded (in the input length) amount of computation time; while the class $\mathsf{EXPSPACE}$, for *exponential space*, contains all tasks that can be solved by an algorithm using an exponentially-bounded amount of memory. One of the central goals of complexity theory is to develop a deep enough understanding of computation to prove *lower bounds* on the amount of resources that are *required* to solve computational tasks. In some instances theorists have been quite successful — it is known that $\mathsf{P} \neq \mathsf{EXPSPACE}$, for example, and so not every problem that can be solved in exponential space can be solved in polynomial time — but many questions (such as the famous P vs. NP problem [20]) remain far out of reach of current techniques.

One of the main lines of research into these lower bound problems is known as *circuit complexity*. The main objects of study here are *boolean circuits*, which are a simple mathematical abstraction of the digital logic circuits present in any computer in the real world. Formally, a boolean circuit C consists of a collection of *wires* which carry boolean (i.e. $\{0, 1\}$) values; these wires connect together *logical gates* which take boolean values as input and then output a boolean value on the gate's output wires. The functions computed at each gate are usually simple (reflecting the simplicity of the logic gates in electronic circuits) — typically we allow the two-input AND (output 1 if both inputs are 1), the two-input OR (output 1 if either input is

1), and the one-input NOT gate (output the opposite of the input). Then, if we designate some subset of wires as "inputs" and another set of wires as "outputs", a boolean circuit becomes a device that computes a boolean function $f : \{0,1\}^{\text{Inputs}} \rightarrow \{0,1\}^{\text{Outputs}}$ in the natural way. Figure 1.1 depicts a simple boolean circuit.

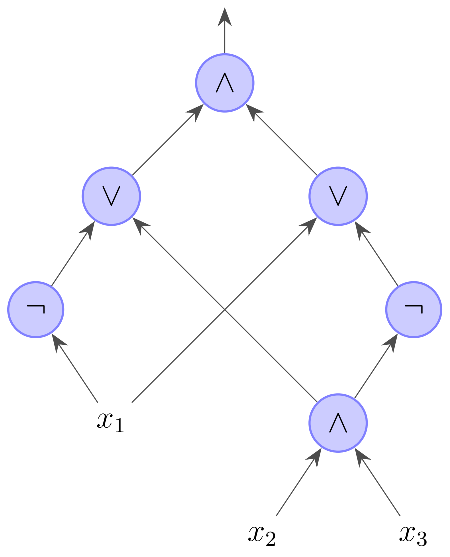

Figure 1.1: A Boolean Circuit. Note $\wedge :=$ AND, $\vee :=$ OR, and $\neg :=$ NOT

From the perspective of complexity theory, boolean circuits are nice to study because they are *combinatorial*. That is, a circuit is "just" a directed acyclic graph with some computational semantics attached to its elements. This is nice as *combinatorial measures* of boolean circuits correspond to computational resources in a nice way: for instance, the *size* (number of gates) of a boolean circuit C measures the amount of computation time it takes to evaluate C on an input if we evaluate each gate in sequence; while the *depth* (longest path from an input wire to an output wire) measures the amount of time it takes to evaluate C when every gate can operate in parallel. This suggests the application of combinatorial techniques to understand circuit complexity, and indeed there have been several outstanding successes along these lines for restricted circuit families. For example:

1. Håstad [30] proved that for every positive integer d, any depth-d circuit C computing the parity of its input bits (i.e. whether the number of 1s is even or odd) requires size $2^{Cn^{1/d-1}}$ for some universal constant C.

2. Håstad [31] proved that for every $\varepsilon > 0$, any boolean formula (that is, a circuit where each gate has exactly one output) performing a certain task in P requires size $n^{3-\varepsilon}$.

3. Razborov [54] proved that any *monotone* circuit (that is, a circuit that does not use NOT

gates, only AND and OR) solving the k-Clique problem (given a graph G, determine if it has a complete subgraph with k nodes) requires size $n^{\Omega(k)}$.

4. Raz and McKenzie [49] gave, for each positive integer i, an example of a computational task that can be efficiently computed by depth-$\log^i n$ monotone circuits, but not by any monotone circuit of depth $\log^{i-1} n$.

The results presented in this thesis are of this type: we prove lower bounds on the sizes of restricted models of boolean circuits performing natural computational tasks. In particular we will focus on *monotone* circuit complexity; we review the details of this theory next.

1.2 Boolean Circuits

Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function on input variables $z_1, z_2, \ldots, z_n$. As we have just outlined, a *boolean circuit* is a computational model used for studying the amount of time it takes to compute f by both parallel and sequential algorithms. Formally, using the shorthand $\wedge \equiv$ AND, $\vee \equiv$ OR, and $\neg \equiv$ NOT, a boolean circuit computing f is defined by a sequence of boolean functions $g_1, g_2, \ldots, g_t$ where

- $g_i = z_i$ for each $i = 1, 2, \ldots, n$,
- $g_t = f$, and
- for each $i = n+1, n+2, \ldots, t$, either $g_i = \neg g_j$ for some $j < i$ or $g_i = g_j \circ g_k$ for some $\circ \in \{\wedge, \vee\}$.

The *depth* of a boolean circuit is the length of the longest path in the corresponding graph, and is a measure of the amount of time it takes to compute f by a parallelized algorithm. This is intuitively justified by thinking of each gate of the circuit as being evaluated by a separate processor, where each processor activates as soon as its inputs are available. On the other hand, the *size* of a boolean circuit is the number of gates, and is a measure of the *sequential* time it takes to compute f. Intuitively, a sequential algorithm can evaluate a boolean circuit by sorting the gates of the circuit in topological order from sources to sink and then processing each gate according to this order. If f is a boolean function then we let $\mathsf{C}(f)$ denote the minimum size of any circuit computing f, and $\mathsf{D}(f)$ denote the minimum depth of any circuit computing f.

Before we continue, a note on conventions. In complexity theory one typically studies the computational resources consumed by an algorithm as a function of its input length. This is modelled as follows: if $\mathbf{f} = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n=1}^{\infty}$ is a sequence of boolean functions (one for each input length), a *circuit family computing* $\mathbf{f}$ is given by a sequence of boolean circuits $\mathbf{C} = \{C_i\}_{i=1}^{\infty}$ where the circuit C_i has i inputs and computes the function f_i. The circuit family is said to be *polynomial-size* if there is a constant c such that $|C_n| = O(n^c)$ for all n. Following convention, we will often be rather loose with this terminology; for example, we use "boolean function" to really mean a sequence of boolean functions and "polynomial-size circuit" to refer to a polynomial-size circuit family.

Continuing on: the class of functions computed by polynomial-size circuit families, denoted P/poly, is an important class in computational complexity theory. It contains every language in P, the class of languages computable by polynomial-time algorithms. However, it is conjectured that P/poly does not contain NP; furthermore, as boolean circuits are generally considered to be easier to analyze than algorithms (usually formalized using Turing Machines), proving lower bounds on the sizes of boolean circuits computing languages in NP is considered to be a promising path to proving $\mathsf{P} \neq \mathsf{NP}$ by many researchers.

With that being said, what are the best known lower bounds on the sizes of boolean circuits? A classic counting argument due to Shannon [59] shows that there are boolean functions on n variables which require very large boolean circuits.

Theorem 1.2.1. *For every sufficiently large positive integer n, there exists a boolean function f on n variables such that $\mathsf{C}(f) \geq \Omega(2^n/n)$.*

Proof Sketch. There are $2^{O(s \log s)}$ boolean circuits with s gates, and exactly 2^{2^n} boolean functions on n variables. Since every boolean circuit computes exactly one function, setting $s = o(2^n/n)$ yields a contradiction. □

This theorem provides very strong — in fact, tight [41] — lower bounds on the size of boolean circuits, but it provides no control over the hard boolean function itself! By applying techniques used in Kannan's theorem [36] one can find a function computable in the (massive) complexity class $\Sigma_2\mathsf{EXP}$ with this large circuit complexity. The best known lower bounds on circuit size for a family of boolean functions that is more efficiently computable is the following lower bound due to Iwama and Morizumi [34].

Theorem 1.2.2. *There exists a family of boolean functions $\{f_n\}_{n=1}^{\infty}$ computable by a polynomial-size boolean circuit family such that for all sufficiently large n, $\mathsf{C}(f_n) \geq 5n - o(n)$.*

The gap between the lower bound in Theorem 1.2.1 and Theorem 1.2.2 is gigantic, and it is a sobering fact that we are still unable to prove superlinear (!) lower bounds on boolean circuits computing functions in NP.

As a result, researchers have focused on studying restricted circuit families, and in this thesis we will focus on *monotone circuits*. Recall that a boolean circuit is monotone if it only contains $\wedge$ and $\vee$ gates (that is, no $\neg$ gates). Monotone boolean circuits get their name since they can only compute *monotone* boolean functions: flipping an input bit from 0 to 1 can not flip the output from 1 to 0 (that is, $x \leq y$ implies $f(x) \leq f(y)$). Since $\wedge$ and $\vee$ are monotone boolean functions, it follows that the function computed by any monotone boolean circuit is also monotone; conversely, every monotone boolean function f is easily seen to be computable by an exponential-size monotone circuit by a truth-table construction.

What are the best lower bounds known for *monotone* boolean circuits? For a monotone boolean function f let $\mathsf{mC}(f)$ denote the size of the smallest monotone circuit computing f, and similarly define $\mathsf{mD}(f)$ to be the minimum depth of a monotone circuit computing f. Pippenger [47] followed the counting argument of Shannon to obtain the following lower bound (observe that it is smaller than the lower bound in Theorem 1.2.1 as there are fewer monotone boolean functions than boolean functions).

Theorem 1.2.3. *For every sufficiently large positive integer n there exists a boolean function f on n variables such that* $\mathsf{mC}(f) \geq \Omega(2^n/n^{3/2})$.

However, unlike the case of non-monotone boolean circuits, we have very strong lower bounds on the size of monotone boolean circuits computing functions in NP. The first super-polynomial lower bounds (on the order of $n^{\Omega(\log n)}$) for a family of monotone boolean circuits computing a boolean function in NP were proved by Razborov [54] for circuits computing the clique function. Andreev [3] proved exponential lower bounds on the order of $2^{n^{1/8-\varepsilon}}$ for every ε for another monotone function in NP, and this lower bound was later sharpened by Alon and Boppana to $2^{\Omega(n^{1/4})}$ [11]. Finally, Harnik and Raz proved a lower bound on the order of $2^{\Omega((n/\log n)^{1/3})}$ [29], which is the strongest lower bound known to date.

1.3 Boolean Formulas

A *boolean formula* is a "tree-like" boolean circuit — the output of each gate can be used at most once. Formally, a formula is defined by a full binary tree T, in which each of the leaves of the tree are labelled with an input variable x_i or its negation $\neg x_i$ and each internal node is labelled with $\wedge$ or $\vee$. A formula is *monotone* if it does not contain any negations, and the *size* of a boolean formula is the number of leaves when it is drawn as a tree. Let $\mathsf{F}(f)$ denote the minimum size of any formula computing f, and similarly define $\mathsf{mF}(f)$ for monotone formula size.

Since a boolean formula is a full binary tree, any depth-d formula has at most 2^d leaves. A remarkable fact (usually attributed to Spira [62], although, it was proved earlier by Khrapchenko [66]) is a converse to the above statement: any formula of size s can be transformed into a formula of depth $O(\log s)$.

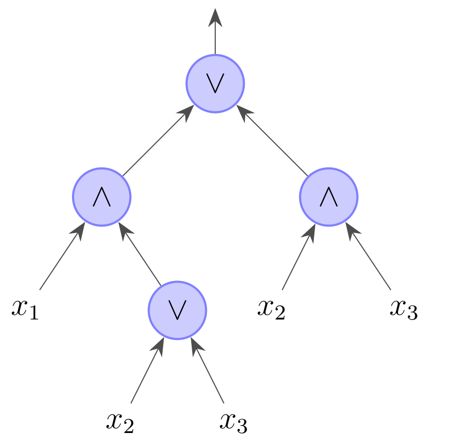

Figure 1.2: A Boolean Formula computing the Majority function.

Theorem 1.3.1 (Formula Balancing Theorem)**.** *For any boolean function f, $\mathsf{D}(f) = \Theta(\log \mathsf{F}(f))$. Furthermore, if f is monotone, then $\mathsf{mD}(f) = \Theta(\log \mathsf{mF}(f))$.*

This implies that the formula size of a boolean function f is a measure of the circuit depth of f and thus the amount of parallel time required to compute f. In complexity theory, the class of languages computable by families of polynomial-size boolean formulas is denoted $\mathsf{NC}^1/\mathsf{poly}$; it is easy to see that $\mathsf{NC}^1/\mathsf{poly} \subseteq \mathsf{P}/\mathsf{poly}$ since every boolean formula is a boolean circuit.

With these results in mind, what are the best lower bounds known for formula size? A counting argument due to Riordan and Shannon [60], analogous to Theorem 1.2.1, yields the following strong lower bound:

Theorem 1.3.2. *For all sufficiently large n there exists a boolean function f on n variables such that $\mathsf{F}(f) \geq \Omega(2^n/\log n)$.*

However, we once again have no control over the “hard” boolean function in this theorem. Unlike the case of boolean circuits, theorists are able to prove super-linear lower bounds on the size of boolean formulas for more natural boolean functions; the best known lower bound for any efficiently computable function is $n^{3-\varepsilon}$ (due to Håstad [31], building on earlier works [4, 33, 46, 63]) for a function computable in P.

Theorem 1.3.3. *For any $\varepsilon > 0$ the following holds: There exists a family of boolean functions $\{f_n\}_{n=1}^{\infty}$ computable in P such that for all sufficiently large n, $\mathsf{F}(f_n) \geq n^{3-\varepsilon}$.*

Once again, in the world of *monotone* formulas we have much better lower bounds. One can mimic the proof of Theorem 1.3.2 to again obtain very strong lower bounds against an “arbitrarily hard” boolean function.

Theorem 1.3.4. *For all sufficiently large n there exists a monotone boolean function f on n variables such that* $\mathsf{mF}(f) \geq \Omega(2^n/\sqrt{n}\log n)$.

Since any monotone boolean formula is also a monotone boolean circuit, it follows that all lower bounds for monotone circuits described in Section 1.2 apply to monotone formulas as well — for instance, we have $2^{\Omega((n/\log n)^{1/3})}$ lower bounds for monotone formulas computing a function in NP [29]. There are a number of other lower bounds on monotone formula complexity — we have exponential separations between monotone formulas and monotone circuits, as well as a stronger lower bound for monotone formulas computing a function in NP. These are reviewed next.

The first superpolynomial lower bounds that separated the power of monotone formulas from monotone circuits were proved by Karchmer and Wigderson [37], who showed that any monotone formula for the *st-connectivity* problem — that is, given a directed graph on n vertices with distinguished vertices s, t, decide if there is a path from s to t — required size $n^{\Omega(\log n)}$. Raz and Wigderson [51] give a stronger lower bound of $2^{\Omega(\sqrt{n})}$ for monotone formulas computing the *matching* problem; however, while the matching problem is computable in P, a result of Razborov [54] shows that it is not computable by polynomial-size monotone circuits. Circumventing this, Raz and McKenzie [49] proved that any monotone formula computing the *GEN* problem — which is also computable by polynomial-size monotone circuits — requires size $2^{\Omega(n^{\varepsilon})}$, where $\varepsilon > 0$ is some universal constant. Moreover, Raz and McKenzie exhibited a *depth-hierarchy theorem* showing, for each positive integer i, a monotone boolean function f_i which is computable by polynomial-size monotone boolean circuits of depth $(\log n)^i$, but such that every monotone boolean circuit of depth $(\log n)^{i-1}$ computing f_i required exponential size. Finally, a recent result of Göös and Pitassi [28] proved $2^{\Omega(n/\log n)}$ lower bounds on monotone formulas computing a monotone function in NP. Despite this progress, note that there is still a large exponential gap between the best lower bounds for monotone formulas computing a function within NP and the lower bounds obtained by counting arguments.

1.4 Switching Networks

A *switching network* is a computational model used for studying the amount of *memory* it takes to compute f by an algorithm. Formally, a switching network is defined by an undirected graph $G = (V, E)$ whose vertices are called *states*. We distinguish two states $s, t \in V$, and each edge $e \in E$ of the network is labelled with an input variable x_i or its negation $\neg x_i$. Given an input assignment $x \in \{0,1\}^n$ to the variables, an edge e is *alive* if its label is satisfied by x; the switching network *accepts* the input x if there is a path of alive edges from s to t. The *size* of a switching network is the number of states in the network. We define $\mathsf{S}(f)$ denote the minimum size of any switching network computing f, and $\mathsf{mS}(f)$ similarly for monotone switching networks.

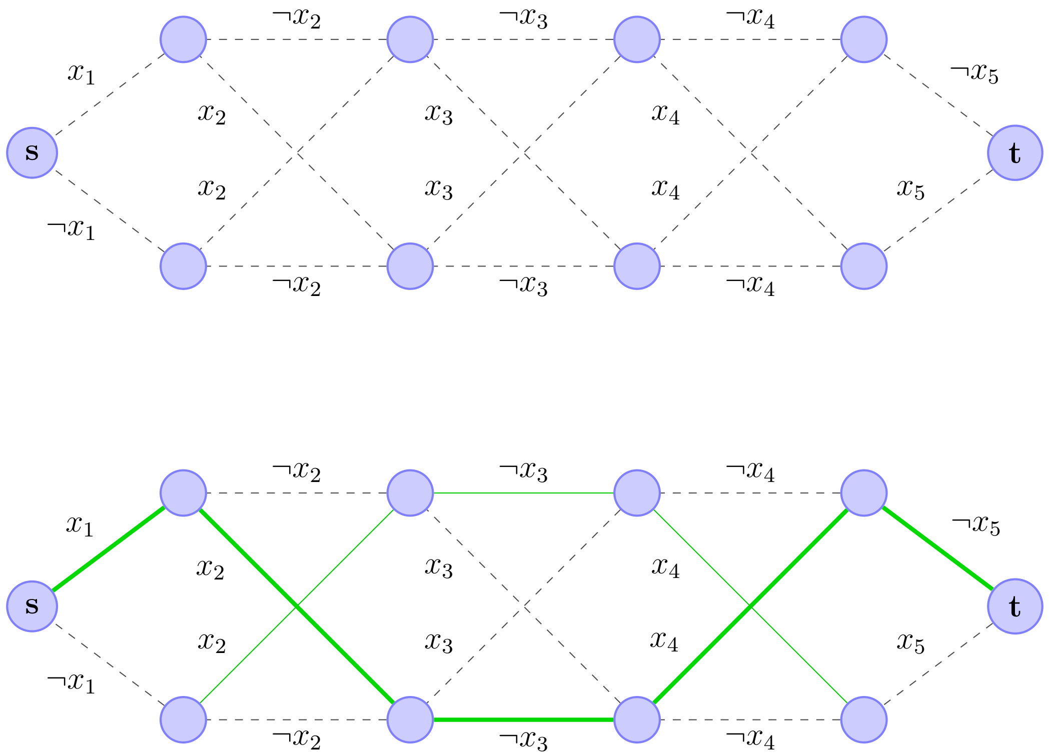

Figure 1.3: A switching network computing the parity of 5 bits, evaluated on $(1, 1, 0, 1, 0)$.

Switching networks were introduced by Shannon [59] as a model of the large telephone switching networks in place at the time, and he showed how one could use them (in principle) to perform computations. The class of boolean functions computable by polynomial-size switching networks is denoted $\mathsf{L/poly}$, and roughly correspond to those languages computable by algorithms using a logarithmic amount of memory. The following result, due to Borodin [12], also relates the size of switching networks to the *circuit depth* of boolean functions.

Theorem 1.4.1. *For any boolean function* f, $\log \mathsf{S}(f) \leq \mathsf{D}(f)$ *and* $\mathsf{D}(f) \leq O(\log^2 \mathsf{S}(f))$. *Both inequalities continue to hold for monotone switching network size and monotone depth.*

It is not hard to simulate a switching network by a boolean circuit, and thus the previous theorem shows $\mathsf{NC}^1/\mathsf{poly} \subseteq \mathsf{L/poly} \subseteq \mathsf{P/poly}$, and further that these inclusions continue to hold if the models are monotone.

What are the strongest lower bounds known for switching networks? A counting argument due to Shannon [59] again yields exponential lower bounds for some (arbitaily hard) boolean function.

Theorem 1.4.2. *For every sufficiently large positive integer n, there exists a boolean function f on n variables such that $\mathsf{S}(f) \geq \Omega(2^n/n)$.*

For non-monotone switching networks, the best lower bound for a function computable in NP (in fact, the function is computable in P) is due to Nečiporuk [44].

Theorem 1.4.3. *There is a family of boolean functions $\{f_n\}_{n=1}^{\infty}$ computable in P such that for all sufficiently large n, $\mathsf{S}(f_n) \geq \Omega((n/\log n)^2)$.*

Also worth mentioning is the lower bound due to Razborov [52] on switching networks computing the Majority function — the technique used by Nečiporuk does not apply to Majority, yet, Razborov was still able to obtain lower bounds of $\Omega(n \log\log\log^* n)$, which is (technically) superlinear.

In the monotone setting, the strongest lower bounds come from applying Theorem 1.4.1 to translate the lower bounds from monotone formulas to monotone switching networks. In particular, applying this theorem to the lower bound of Göös and Pitassi [28] yields $2^{\Omega(\sqrt{n/\log n})}$ lower bounds for monotone switching networks computing a monotone function in NP. In a separate, but elegant, line of work, Potechin [48] proved $n^{\Omega(\log n)}$ lower bounds for switching networks computing the directed st-connectivity function; a nice corollary is that the $O(\log^2 \mathsf{S}(f))$ factor is tight in Theorem 1.4.1 in the monotone case. Potechin's result was improved by Chan and Potechin [18], who obtained exponential lower bounds for monotone switching networks computing the GEN function (which is computable by polynomial-size monotone circuits) and also the clique function.

1.5 Span Programs

Let $\mathbf{F}$ be a field. An $\mathbf{F}$-*span program* is an exotic device for computing boolean functions using linear algebra over $\mathbf{F}$. Formally, a span program is defined by a matrix A over $\mathbf{F}$ whose rows are labelled by boolean literals over variables $z_1, z_2, \ldots, z_n$. Given a span program A, a row vector A_i of A is *consistent* with an input $z \in \{0,1\}^n$ if the literal labelling A_i evaluates to 1 under z. The span program A then *accepts* an input assignment $z \in \{0,1\}^n$ if the set of rows consistent with z span the all-1s vector; with this definition a span program A computes a boolean function $f : \{0,1\}^n \to \{0,1\}$ in the natural way. Said another way: a span program A associates a vector space $V(\ell)$ with each input literal ℓ over $\{z_1, z_2, \ldots, z_n\}$. Given an input assignment to the variables z, the span program *accepts* z if and only if the closure of the vector spaces consistent with z contains the all-1 vector. A span program is depicted in Figure 1.4, where it is also evaluated on the input $(1,0,0)$ — adding the "x_1 row" to the "$\neg x_3$ row" yields the all-1s vector, so we accept.

The *size* of the span program is the number of rows of A. A span program is *monotone* if all literals labelling rows of A are positive, and note that monotone span programs compute monotone functions since adding row vectors can only increase the linear span.

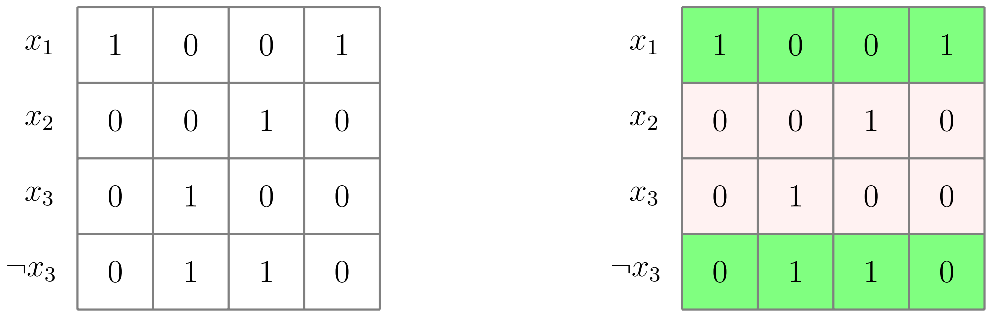

Figure 1.4: A span program, evaluated on $(1, 0, 0)$.

Span programs were introduced by Karchmer and Wigderson [38], who showed that span programs over fields of non-zero characteristic p compute exactly the languages in the complexity class $\mathsf{Mod}_p\mathsf{L}/\mathsf{poly}$, which roughly correspond to counting accepting paths in a non-deterministic, log-space Turing Machine modulo p. This complexity class is rather exotic, but has a nice expression in terms of *directed* switching networks. A directed switching network is simply a switching network with directed edges instead of undirected edges; for any field $\mathbf{F}$ of positive characteristic define an $\mathbf{F}$-directed switching network to be a switching network which accepts an input z if and only if the number of live paths from s to t on input z is congruent to 0 in $\mathbf{F}$. Karchmer and Wigderson [38] proved that $\mathbf{F}$-span programs and $\mathbf{F}$-directed switching networks can efficiently simulate one another; furthermore, they showed that span programs over *any* field can efficiently simulate switching networks (proving $\mathsf{L}/\mathsf{poly} \subseteq \mathsf{Mod}_p\mathsf{L}/\mathsf{poly}$ for every characteristic p).

What lower bounds are known for span programs? Once again, in the non-monotone world our lower bounds are weak if we require the boolean functions to be computable in a small complexity class. As usual, an adaptation of the counting arguments apply to span programs as well: over $\mathrm{GF}(2)$, this counting argument was already exhibited by Nečiporuk [43], who proved the following lower bound using the model of $\mathrm{GF}(2)$-directed switching networks.

Theorem 1.5.1. *For all sufficiently large n, there exists a boolean function f on n variables that requires $\mathrm{GF}(2)$-span programs of size $\sqrt{2^{n+1}}$.*

Karchmer and Wigderson [38] were able to adapt the lower bound of Razborov [52] to prove superlinear lower bounds for non-monotone $\mathrm{GF}(2)$-span programs computing Majority, which is obviously efficiently computable.

Theorem 1.5.2. *Any* $\mathrm{GF}(2)$*-span program computing the Majority function on* n *bits requires size* $\Omega(n \log\log\log^* n)$.

What is known about the complexity of *monotone* span programs? Well, an interesting feature of monotone span programs is that they are not "really" monotone — monotone span programs use non-monotone operations to compute monotone functions. Largely because of this, the relationship between monotone span programs and monotone circuits has been unresolved! It is known that monotone span programs can be much more powerful than monotone circuits: Babai et al. [5] exhibited a function with linear size monotone span programs that requires superpolynomial-size monotone circuits and exponential-size monotone formulas. This immediately implies that the size and depth lower bound methods for monotone circuits cannot be used to prove lower bounds for monotone span programs.

Perhaps because of this, there has been a long and interesting history of monotone span program lower bounds. Karchmer and Wigderson [38], showed that all threshold functions over $\mathrm{GF}(2)$ require monotone span programs of size $\Omega(n \log n)$, which was quickly improved by Csirmaz [23] to an $\Omega(n^2/\log n)$ lower bound. Beimel et al. [9] gave a lower bound of $n^{5/2}$ for a monotone function in P, and then Babai et al. [5] improved their technique, obtaining the first superpolynomial lower bound of $n^{\Omega(\log n/\log\log n)}$ for a monotone function in NP. Each of these results were obtained by direct combinatorial arguments, which were simplified and improved by Gál to $n^{\Omega(\log n)}$ [25]. The superpolynomial lower bounds cited above only applied to functions computable in NP, so, to improve this Beimel and Weinreb [10] gave quasipolynomial lower bounds $n^{\Omega(\sqrt{\log n})}$ for a function in P, proving that monotone span programs can be weaker than polynomial time, as well as separating span programs over different fields. Prior to the results of this thesis, there were *no* exponential lower bounds for monotone span programs other than the bounds obtained by counting arguments.

Linear Secret Sharing Schemes. Closely related to monotone span programs are *secret sharing schemes*, which are a simple cryptographic device defined as follows. We have a "dealer" who has some "secret" (say, an element k of a field $\mathbf{F}$), a set of n parties, and an upward-closed collection $\mathcal{A} \subseteq 2^{[n]}$ of subsets of the n parties called an *access structure*. A *secret sharing scheme* for $\mathcal{A}$ is a method of sharing information with the n parties such that any set of parties in $\mathcal{A}$ can reconstruct the dealer's secret, while any subset of parties not in $\mathcal{A}$ are unable to reconstruct the secret. For the sake of completeness we record the definition of secret sharing schemes here and refer the interested reader to [8] for further details.

Definition 1.5.3. A *distribution scheme* over a domain K is a pair $\Sigma = (\Pi, \mu)$ where μ is a probability distribution over a set R and Π maps pairs in $K \times R$ to tuples $K_1 \times K_2 \times \cdots \times$

K_n, where K_j is the *domain of shares* of player p_j. Given a distribution scheme Σ, a dealer distributes a secret $k \in K$ to n players as follows: first, the dealer samples a random string $r \in R$ and computes $\Pi(k, r) = (s_1, s_2, \dots, s_n)$. Then for each $i \in [n]$, the dealer privately communicates share s_i to the i^{th} player. A distribution scheme is a *secret sharing scheme* for an access structure $\mathcal{A} \subseteq 2^{[n]}$ if it satisfies the following two properties:

Perfect Reconstruction. The secret can be reconstructed by any set of parties in the access structure, i.e. for any set of parties $A \in \mathcal{A}$ there exists a mapping $R_A : \prod_{i \in A} K_i \to K$ such that for every $k \in K$, $\Pr[R_A(\Pi(k, r)_A) = k] = 1$.

Perfect Privacy Every unauthorized set cannot learn anything from their shares (in the statistical sense). In other words, for any $B \notin \mathcal{A}$, for every pair of secrets $k_1, k_2 \in K$, and every vector of shares $v(s_i)_{i \in B}$ we have $\Pr[\Pi(k_1, r) = v] = \Pr[\Pi(k_2, r) = v]$.

The *information ratio* of a distribution scheme is $\max_{1 \leq j \leq n} \log |K_j| / \log |K|$, and measures the relative amount of information shared between parties. A secret sharing scheme is *linear* over a field $\mathbf{F}$ if $K = \mathbf{F}$, the random strings are field elements chosen uniformly random from $\mathbf{F}$, and the shares are vectors over $\mathbf{F}$ chosen by taking linear combinations of the secret and the random strings.

Linear secret sharing schemes are an important subclass of secret sharing schemes as many of the schemes from the literature turn out to be linear. Karchmer and Wigderson [38] proved that monotone span programs over a finite field $\mathbf{F}$ of size s for a monotone function $f : \{0,1\}^n \to \{0,1\}$ imply secret sharing schemes with information ratio s for the natural access structure associated with f (take all subsets $A \subseteq [n]$ such that $f(A) = 1$, using the set-theoretic notation for boolean functions). Conversely, Beimel [7] showed that linear secret sharing schemes imply monotone spans programs, and thus the two objects are equivalent.

Theorem 1.5.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function and let $\mathcal{A}_f$ denote the related access structure. For any finite field $\mathbf{F}$ there exists a monotone span program for f of size s if and only if there exists a linear secret sharing scheme for $\mathcal{A}_f$ over $\mathbf{F}$ with information ratio s.*

This theorem gives a natural alternative motivation for studying monotone span program lower bounds.

1.6 Comparator Circuits

A simple circuit element which is common in digital logic is the *comparator gate*: when given a pair of bits, it outputs them in sorted order. A *comparator circuit* is a boolean circuit made completely of comparator gates, and are the boolean analogue of *sorting networks*, which are models of input-oblivious sorting algorithms. Shallow sorting networks have many applications in theoretical computer science, and explicit constructions have been extensively studied in the literature (see [40] for a survey); the famous AKS construction gives $O(\log n)$ depth sorting networks [1], and an alternative construction was recently given by Goodrich [26].

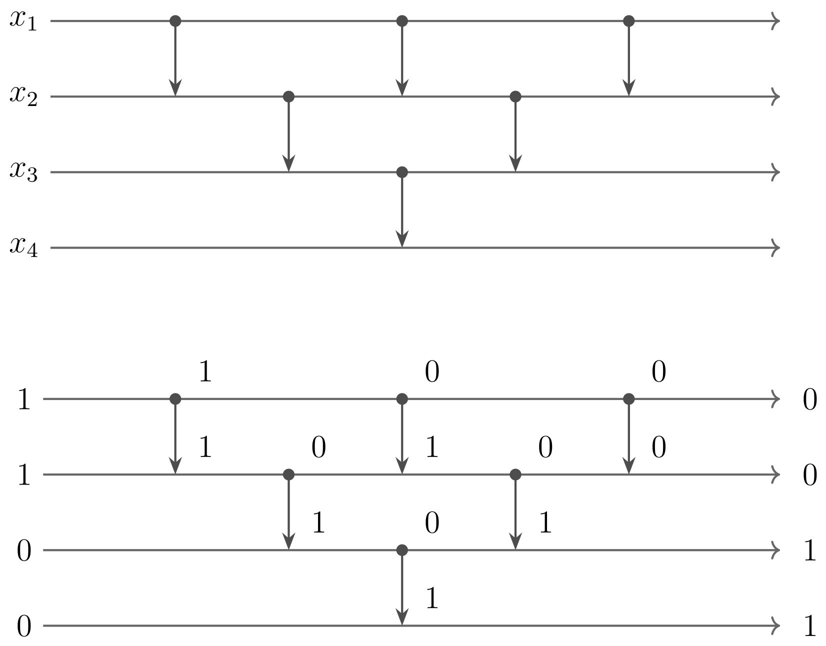

Figure 1.5: A comparator circuit, evaluated on the input $(1, 1, 0, 0)$.

To be more formal, a *comparator circuit* consists of a set of m wires and a sequence $(i_1, j_1), (i_2, j_2), \ldots, (i_s, j_s)$ of comparator gates, each connecting a pair of wires (in this notation, the $\wedge$ output of the comparator gate is attached to the first wire, and the $\vee$ output of the comparator gate is attaced to the second wire). The *size* of the circuit is m, the number of wires, and each wire is initially labelled with either 0, 1, or a boolean literal. We will be interested in comparator circuits which compute boolean functions $f : \{0, 1\}^n \to \{0, 1\}$, where n is possibly less than the number of wires; so, we designate one of the wires as the output wire. A comparator circuit C is *monotone* if no input wire of C is labelled with the negation of an input variable, and it is clear from the monotonicity of comparator gates that monotone comparator circuits compute only monotone functions.

It is not hard to see that comparator circuits are incapable of copying bits in intermediate computations since the hamming weight of the output of each comparator gate is the same as the hamming weight of the input. The other notable class of circuits which can not re-use intermediate computations are *boolean formulas*; due to this comparator circuits have been used as a method of studying the power of copying bits "beyond" boolean formulas [64]. The class of problems computable by polynomial-size comparator circuits is called $\mathsf{CC}/\mathsf{poly}$, and the structural complexity of this class was recently studied by Cook, Filmus and Le [22]. Despite their inability to copy, polynomial-size comparator circuits are surprisingly powerful: they can compute everything in non-deterministic log-space and appear to be incomparable with poly-logarithmic depth circuits [22, 64].

What is known about lower bounds for comparator circuits? One can mimic the counting argument as before to get an exponential lower bound:

Theorem 1.6.1. *For all sufficiently large positive integers n, there is a boolean function f on n variables such that every comparator circuit computing f requires $\Omega(\sqrt{2^n/n})$ wires.*

However, to the best of our knowledge, there are no other lower bounds known for non-monotone comparator circuits, other than the bounds known for unrestricted boolean circuits. Indeed, the situation is the same for *monotone* comparator circuits: all lower bounds in the literature follow directly from the lower bounds for monotone boolean circuits.

1.7 Our Contribution.

While monotone circuit complexity is quite well-developed when compared to non-monotone circuit complexity, several significant open problems remain (particularly, in the complexity of monotone span programs). We collect these problems below.

1. By counting arguments one can show that almost all monotone functions on n variables require size $2^{\Omega(n)}$ in *all* of the models above; however, *we can not prove such a lower bound for* any *of these models for a function computable in* NP*!*

2. The strongest lower bounds on the size of monotone span programs (over *any* field) that do not use counting arguments are $n^{\Omega(\log n)}$ for a monotone function computable in NP. For *all* the other circuit models we listed above exponential lower bounds are known. *Can we prove exponential lower bounds for monotone span programs?*

3. Babai, Gál and Wigderson showed that there is a function computable by polynomial-size monotone span programs over $GF(2)$ that requires monotone circuits of size $n^{\Omega(\log n)}$ [5], no separation is known in the other direction for *any* field. Such a separation is already known for both monotone formulas and monotone switching networks (and, note that monotone span programs over *any* field can simulate both of these models). *Can monotone circuits be more powerful than monotone span programs?* One can ask the same question for monotone comparator circuits: *Are monotone circuits more powerful than monotone comparator circuits?*

4. Is there a monotone function computable by polynomial-size *non-monotone* span programs that requires large *monotone* span programs? Equivalently: *is the monotone restriction significant for the efficiency of span programs?* One can ask the same question for comparator circuits: *is the monotone restriction significant for the efficiency of comparator circuits?*

5. For any two fields $\mathbf{F}, \mathbf{F}'$ of different characteristic, Beimel and Weinreb [10] exhibited a boolean function that is computable by polynomial-size monotone span programs over $\mathbf{F}$, but any monotone span program over $\mathbf{F}'$ requires size $n^{\Omega(\sqrt{\log n})}$. *Can this separation be improved?*

We resolve all of these problems, and do so in a unified way — the solution to each of these problems will be a consequence of a single theorem that we prove later in the thesis (cf. Theorem 5.0.1). Furthermore, our general theorem can be used to provide alternative proofs of many of the known lower bounds discussed above, such as the lower bounds against monotone formulas by Karchmer and Wigderson [37], the depth hierarchy theorem of Raz and McKenzie [49], and the lower bounds against monotone switching networks by Potechin and Chan-Potechin [18, 48].

What techniques can we use to prove such a theorem? At a high level, our approach is to use ideas from *hardness escalation*, which is a rapidly growing body of techniques in complexity theory [19, 27, 28, 32, 39, 49]. The basic idea of hardness escalation is very simple: reduce the study of algorithms in a "powerful" model of computation to the study of algorithms in a "weak" model of computation. To do so, one typically studies algorithms for "structured" functions in the "powerful" model of computation, which can then be reduced to studying "simple" functions in the "weak" model of computation.

We will prove a hardness escalation result for a simple matrix-theoretic complexity measure introduced by Razborov [55]. For any field $\mathbf{F}$, he defined a measure of boolean functions $\mu_{\mathbf{F}}(f)$ (which we will call the *rank measure*) to study lower bounds on formula size for boolean functions f; using this measure Razborov gave a simple proof that any monotone formula computing a certain monotone function in NP must have size at least $n^{\Omega(\log n)}$. While not the strongest lower bound known against monotone formula size — similar bounds were already known for st-connectivity [37], and stronger lower bounds are known for other functions [28, 50] — Razborov's method is exceptionally elegant, and further research showed that $\mu_{\mathbf{F}}(f)$ is, in fact, a lower bound on monotone span programs, monotone formulas, monotone switching networks, and monotone comparator circuits! So, for the purposes of lower bounds we will prove a hardness escalation-type theorem for $\mu_{\mathbf{F}}(f)$.

The starting place for our theorem is the *search problem* associated with unsatisfiable CNF formulas. More formally, associated with any unsatisfiable CNF formula $\mathcal{C}$ is the following search problem $\mathsf{Search}(\mathcal{C})$: given an assignment to the variables of $\mathcal{C}$, output any clause in $\mathcal{C}$ falsified by the assignment. Given an unsatisfiable CNF $\mathcal{C}$, there is a natural method (introduced by Raz and McKenzie [49] and refined by both Göös and Pitassi [28] and Oliveira [45]) of associating a "lifted function" $f_{\mathcal{C}}$ with $\mathcal{C}$ that can be thought of as a monotone variant of the SAT problem. With this in hand, we introduce a new complexity measure on $\mathsf{Search}(\mathcal{C})$ that we call the *algebraic gap complexity*, and show how to translate lower bounds on the algebraic gap complexity of $\mathcal{C}$ into lower bounds against the rank measure $\mu_{\mathbf{F}}(f_{\mathcal{C}})$. This allows us to resolve the questions posed above by proving the appropriate bounds on algebraic gaps for certain special formulas $\mathcal{C}$.

Now, rather than *directly* prove lower bounds on the algebraic gap complexity, we instead show that the algebraic gap of $\mathcal{C}$ is *exactly* the same as the *Nullstellensatz degree* of refuting $\mathcal{C}$, which is a very well-studied measure in *propositional proof complexity*. This provides a new and interesting dual characterization of Nullstellensatz degree, and also allows us to use the broad literature on Nullstellensatz degree lower bounds [6,13–15,57] in a black-box fashion to prove our results. Already, the techniques we have discussed are sufficient to resolve problems 1 through 4.

To resolve Problem 5 we prove a second hardness escalation theorem, this time just for span programs. Gál [25] introduced an interesting algebraic measure on monotone boolean functions that she calls the *algebraic tiling number*, denoted $\chi_{\mathbf{F}}(f)$, and shows that the algebraic tiling number is exactly the monotone span program size for computing f. In a separate argument from above, we show how to translate Nullstellensatz degree *upper bounds* on an unsatisfiable CNF $\mathcal{C}$ into upper bounds on the size of the algebraic tiling number $\chi_{\mathbf{F}}(f_{\mathcal{C}})$ (and thus also for monotone span program size). By combining this with the lower bound discussed above we obtain a *characterization* of the size of monotone span programs for lifted functions $f_{\mathcal{C}}$ in terms of the Nullstellensatz degree of the underlying CNF $\mathcal{C}$. Then, by exploiting known separations between Nullstellensatz degree over different fields [15] we resolve Question 5.

Finally, in the process of proving our lifting theorems, we prove a technical theorem that we hope will be of independent interest, showing how to transform an $\mathbf{F}$-valued multilinear polynomial p into a matrix M_p such that the *rank* of the matrix M_p can be calculated directly from the set of monomials occurring in p. In particular, the statement of our theorem is a generalization of a similar "degree-to-rank" connection exhibited by Sherstov [61] for real polynomials; however, our theorem is far more general.

The rest of this thesis is organized as follows:

Chapter 2. Preliminaries. We introduce several basic concepts that will be used throughout the thesis, such as the Karchmer-Wigderson games [37], Razborov's rank measure [55], and the algebraic tiling number [25]. We show that the rank measure $\mu_{\mathbf{F}}(f)$ lower bounds the size of monotone formulas, monotone switching networks, monotone span programs, and monotone comparator circuits — this was shown for formulas by Razborov [55], for span programs and switching networks by Gál [25], and showing $\mu_{\mathbf{F}}(f)$ is a lower bound on monotone comparator circuits is an original contribution of this thesis.

Chapter 3. Pattern Matrices and the Rank Measure. We introduce one of the main technical tools (*pattern matrices*) that we use to analyze the rank measure. Using pattern matrices, we prove a "warmup" version of our main lifting theorem, reducing the *real* rank measure $\mu_{\mathbb{R}}$ to the *real* algebraic gap complexity.

Chapter 4. Algebraic Gaps and Nullstellensatz. Here we study the algebraic gap complexity more deeply. In particular, we show it is exactly the same as the classical *Nullstellensatz degree*, which is a complexity measure of propositional tautologies studied in *propositional proof complexity*. The results in this chapter allow us to appeal to the vast literature of Nullstellensatz degree bounds when trying to lower bound the algebraic gap complexity.

Chapter 5. Main Lifting Theorems. In this chapter we give a simple analysis of the rank of pattern matrices. Then, using the tools we have developed, we prove our main lifting theorems connecting the algebraic gap complexity and the Nullstellensatz degree to the rank measure and algebraic tiling number.

Chapter 6. Applications. We apply our lifting theorems to obtain a number of lower bounds in monotone complexity theory, resolving problems 1 through 5 and unifying the proofs of many prior lower bounds in the literature.

Chapter 7. Conclusion. We conclude with a collection of open problems.

The results of this thesis appear in the following three papers:

1. Robert Robere, Toniann Pitassi, Benjamin Rossman and Stephen A. Cook. *Exponential lower bounds for monotone span programs.* In Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS 2016). 406-415, 2016.

2. Toniann Pitassi, Robert Robere. *Strongly exponential bounds lower bounds for monotone computation.* In the Proceedings of the 49th Annual Symposium on Theory of Computing (STOC 2017). 1246-1255, 2017.

3. Toniann Pitassi, Robert Robere. *Lifting Nullstellensatz to Monotone Span Programs over any Field.* To appear in the Proceedings of the 50th Annual Symposium on Theory of Computing (STOC 2018).

Chapter 2

Preliminaries

Let $\mathcal{Z}$ be a set and let n be a positive integer. We use the standard notation of $\mathcal{Z}^n$ to represent the set of all n-tuples over $\mathcal{Z}$. If $z \in \mathcal{Z}^n$ then z_i denotes the ith element of the tuple z, and if $A \subseteq [n]$ then z_A is the tuple of elements in z indexed by A. If $A \subseteq [n]$ we will often write things like $z = z_A z_{[n]\setminus A}$ to mean the natural partition of z into the tuples indexed by A and $[n] \setminus A$, even though the indices of A may not technically be the "first" elements of the tuple z.

Let $\mathbf{F}$ be a field. If $\mathcal{X}, \mathcal{Y}$ are (ordered) sets then we consider functions $A : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ as $|\mathcal{X}| \times |\mathcal{Y}|$ matrices over $\mathbf{F}$, where the rows of A are indexed by elements of $\mathcal{X}$ and columns of A are indexed by $\mathcal{Y}$. To simplify notation, we refer to such a function A as an $\mathcal{X} \times \mathcal{Y}$ matrix over $\mathbf{F}$, and use regular function notation (e.g. $A(x, y)$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$) to index into such matrices. Let $\mathbb{1}_{\mathcal{X},\mathcal{Y}}$ denote the $\mathcal{X} \times \mathcal{Y}$ all-$1s$ matrix, but we will often leave out the subscript if the dimensions of the matrix are clear from the context.

If $\mathcal{X}, \mathcal{Y}$ are sets then a *combinatorial rectangle* in $\mathcal{X} \times \mathcal{Y}$ is a subset $R \subseteq \mathcal{X} \times \mathcal{Y}$ such that $R = X \times Y$ for some subsets $X \subseteq \mathcal{X}, Y \subseteq \mathcal{Y}$. If $A : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ is a matrix and R is a rectangle in $\mathcal{X} \times \mathcal{Y}$ then let $A\restriction R$ denote the submatrix of A indexed by elements of R. It will be convenient to think of $A\restriction R$ as having the same dimensions as A, and thus we formally define $A\restriction R : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ by

$$(A\restriction R)(x, y) = \begin{cases} A(x, y) & \text{if } (x, y) \in R \\ 0 & \text{otherwise.} \end{cases}$$

The matrix A is *embedded* in R if $A = A\restriction R$ — i.e. all non-zero entries of A are indexed by R.

2.1 Boolean Functions

A *boolean function* is a function mapping strings of bits to bits, i.e. a function of the form $f : \{0, 1\}^n \to \{0, 1\}$ for some non-negative integer n. A *restriction* of f is a map $\pi : [n] \to$

$\{0, 1, *\}$, which should be thought of as setting some of the inputs to f to $\{0, 1\}$ values while leaving other values (the $*$s) unset. If π is a restriction then $\mathsf{vars}(\pi) = \pi^{-1}(\{0, 1\})$ is the set of inputs restricted by π, and $f\restriction\pi : \{0, 1\}^{n-|\mathsf{vars}(\pi)|} \to \{0, 1\}$ is the function obtained from f by applying the restriction to the inputs of f.

If $x, y \in \{0, 1\}^n$ then $x \leq y$ if $x_i \leq y_i$ for all i and $x < y$ if $x \leq y$ and $x \neq y$. A boolean function f is *monotone* if $x \leq y$ implies that $f(x) \leq f(y)$ for all x, y. Fix a monotone boolean function f. An input $x \in \{0, 1\}^n$ is a *minterm* of f if $f(x) = 1$ and $f(x') = 0$ for all $x' < x$. Dually, an input $y \in \{0, 1\}^n$ is a *maxterm* if $f(y) = 0$ and $f(y') = 1$ for all $y' > y$. Let $\mathcal{M} \subseteq f^{-1}(1)$ be the set of all minterms of f; it is clear that $\mathcal{M}$ forms an antichain (for all distinct $x, y \in \mathcal{M}$ neither $x \geq y$ nor $x \leq y$ holds) in the poset $(\{0, 1\}^n, \leq)$. It is clear that the set of minterms *defines* the function f; this will be useful later so we record it as a proposition.

Proposition 2.1.1. *Let $f : \{0, 1\}^n \to \{0, 1\}$ be a monotone boolean function and let $\mathcal{M}$ be the set of minterms of f. Let $g : \{0, 1\}^n \to \{0, 1\}$ be a boolean function defined by $g(x) = 1$ if and only if $x \geq y$ for some $y \in \mathcal{M}$. Then $g = f$.*

Proof. Suppose otherwise, and let x be an input such that $g(x) \neq f(x)$. Clearly x can not be a minterm of f. If $g(x) = 1$ then there is a $y \in \mathcal{M}$ such that $x \geq y$; since y is a minterm of f and f is monotone it follows that $f(x) = 1$, a contradiction. On the other hand, if $g(x) = 0$ then $f(x) = 1$. Let $y \leq x$ be any minimal input such that $f(y) = 1$; clearly y is a minterm and $x \geq y$, contradicting that $g(x) = 0$. $\square$

A *partial* boolean function is a function $f : \{0, 1\}^n \to \{0, 1, *\}$ (informally, if $f(x) = *$ then we "don't care" what the output of the function is); a partial boolean function f is *total* if $\mathrm{range}(f) = \{0, 1\}$. A partial boolean function $f : \{0, 1\}^n \to \{0, 1, *\}$ is *monotone* if it can be extended to a total monotone boolean function by choosing $\{0, 1\}$-assignments for the $*$ outputs. A boolean circuit model (such as any circuit model considered in the introduction) computes a partial boolean function f if it agrees with the $\{0, 1\}$-outputs of f.

Suppose $f : \{0, 1\}^n \to \{0, 1\}$ and $g : \{0, 1\}^m \to \{0, 1\}$ are boolean functions on variables $y_1, y_2, \ldots, y_n$ and $z_1, z_2, \ldots, z_m$, respectively. Let $\rho : \{y_1, y_2, \ldots, y_n\} \to \{z_1, z_2, \ldots, z_m, 0, 1\}$ be a map from the variables of f to the variables of g and constants $\{0, 1\}$. Under such a map ρ we can consider $g(\rho(\cdot))$ as a function on the variables of f; the mapping ρ is called a *monotone projection* if $g(\rho(y)) = f(y)$ for all $y \in \{0, 1\}^n$. A monotone projection is one of the simplest form of reducibility between functions: it says that we can "compute" f using g just by relabelling variables and plugging in constants.

2.2 Karchmer-Wigderson Games

Rather than study monotone circuit models directly, we instead study the following combinatorial search problem introduced by Karchmer and Wigderson [37].

Definition 2.2.1. Let $f : \{0,1\}^n \to \{0,1,*\}$ be a partial boolean function and let $\mathcal{U} = f^{-1}(1), \mathcal{V} = f^{-1}(0)$. The *Karchmer-Wigderson game* of f is the following search problem KW_f: given as input a pair $(u,v) \in \mathcal{U} \times \mathcal{V}$, output an index $i \in [n]$ such that $u_i \neq v_i$.

If f is monotone, then further define the *monotone Karchmer-Wigderson game* to be the following search problem mKW_f: given as input a pair $(u,v) \in \mathcal{U} \times \mathcal{V}$, output an index $i \in [n]$ such that $u_i = 1$ and $v_i = 0$.

Karchmer and Wigderson [37] introduced their games to apply tools from *communication complexity* to problems in circuit complexity. As an example, let us consider solving KW_f by a deterministic communication protocol. There are two players, Alice and Bob; Alice receives an input $u \in \mathcal{U}$ and Bob receives an input $v \in \mathcal{V}$. Their goal is to agree on an index $i \in [n]$ such that $u_i \neq v_i$. In order to do this they are allowed to communicate bits to each other over a communication channel. They meet before seeing their inputs and agree on a *communication protocol* that will correctly solve the search problem on arbitrary inputs $(u,v) \in \mathcal{U} \times \mathcal{V}$. The *(deterministic) communication complexity* of mKW_f is then the minimum number of bits which must be communicated by Alice and Bob in any communication protocol that solves mKW_f. Karchmer and Wigderson [37] showed that this is exactly the same as the *minimum depth* of any monotone circuit computing the function f.

Theorem 2.2.2. *Let f be any partial boolean function. The deterministic communication complexity of KW_f is exactly $\mathsf{D}(f)$, and similarly the deterministic communication complexity of mKW_f is exactly $\mathsf{mD}(f)$.*

Karchmer and Wigderson were then able to prove $\Omega(\log^2 n)$ bounds on the monotone Karchmer-Wigderson game associated with the *st-connectivity* problem [37]; applying Theorems 2.2.2 and 1.3.1 yields $n^{\Omega(\log n)}$ lower bounds on monotone formula size, as mentioned in Section 1.3.

Karchmer-Wigderson games have become one of the central tools for studying circuit complexity and, in particular, *monotone* circuit complexity — for instance, *all* of the lower bounds on monotone formula size discussed at the end of Section 1.3 were proved by analyzing the corresponding communication protocol. Researchers have discovered many other complexity measures of Karchmer-Wigderson games that characterize the complexity of computing boolean functions in other circuit models. Over the next several sections we will discuss several useful results and complexity measures of Karchmer-Wigderson games that will be studied in this thesis.

2.3 Monotone Karchmer-Wigderson and Rectangle Covers

The monotone Karchmer-Wigderson game mKW_f have a natural characterization in terms of *rectangle covers*, and furthermore this seems to be the "right" level of abstraction in which to study them.

Definition 2.3.1. Let $f : \{0,1\}^n \to \{0,1,*\}$ be a partial monotone boolean function with $\mathcal{U} = f^{-1}(1)$ and $\mathcal{V} = f^{-1}(0)$. For any input $i \in [n]$ define the *coordinate rectangle at* i to be $X_i = \{u \in \mathcal{U} \mid u_i = 1\} \times \{v \in \mathcal{V} \mid v_i = 0\} \subseteq \mathcal{U} \times \mathcal{V}$.

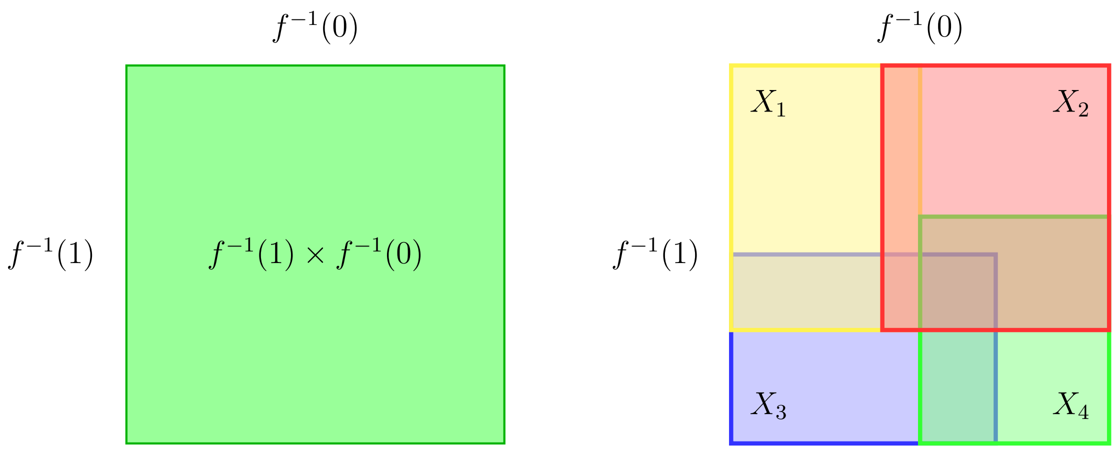

Figure 2.1: A Monotone Karchmer-Wigderson Game as a Rectangle Covering

Two immediate observations about each coordinate rectangle X_i: first, X_i is the set of all pairs of inputs (u, v) on which $i \in [n]$ is a valid output in the monotone Karchmer-Wigderson game; second, X_i is (trivially) a combinatorial rectangle in $\mathcal{U} \times \mathcal{V}$. Since the mKW_f game is *total* (i.e. every input $(u, v) \in [n]$ has a valid output), it follows that the collection of rectangles $\{X_i\}_{i\in[n]}$ covers every entry in $\mathcal{U} \times \mathcal{V}$. This suggests the following definition.

Definition 2.3.2. Let $\mathcal{X}, \mathcal{Y}$ be sets. A *rectangle covering* of $\mathcal{X} \times \mathcal{Y}$ is a collection $\mathcal{R}$ of combinatorial rectangles in $\mathcal{X} \times \mathcal{Y}$ such that every element $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is contained in some rectangle in $\mathcal{R}$. The *size* of the rectangle covering is $|\mathcal{R}|$.

As we have observed, if f is a monotone boolean function on n variables then the monotone Karchmer-Wigderson game of f yields a natural rectangle covering of $\mathcal{U} \times \mathcal{V}$ by the coordinate rectangles of mKW_f; from now on we will sometimes abuse notation and use mKW_f to refer both to the search problem and to the corresponding rectangle covering of $\mathcal{U} \times \mathcal{V}$ by coordinate rectangles. The main observation of this section is a *converse*: if $\mathcal{X}, \mathcal{Y}$ are sets and $\mathcal{R}$ is a rectangle covering of $\mathcal{X} \times \mathcal{Y}$ by n rectangles, then there is a monotone boolean function f on n variables such that mKW_f is the *same* rectangle covering as $\mathcal{R}$, up to renaming the elements of $\mathcal{X} \times \mathcal{Y}$. We can use this result to "forget" about working with fixed monotone boolean functions and instead work with arbitrary rectangle covers $\mathcal{R}$, which we can freely convert into its corresponding monotone boolean function when needed. This proposition is, in some sense, the key step which makes monotone Karchmer-Wigderson games easy to analyze compared to their non-monotone counterparts.

Proposition 2.3.3 (Folklore, [25, 55])**.** *Let $\mathcal{X}, \mathcal{Y}$ be sets and let $\mathcal{R}$ be a rectangle covering of $\mathcal{X} \times \mathcal{Y}$. Then there exists a partial monotone boolean function f on $|\mathcal{R}|$ variables such that mKW_f, viewed as a rectangle covering of $f^{-1}(1) \times f^{-1}(0)$, is equivalent to $\mathcal{R}$ up to the relabelling.*

Proof. Write $\mathcal{R} = \{R_i\}_{i=1}^n$, where $R_i = S_i \times T_i$ for each $i \in [n]$. For each $x \in \mathcal{X}$ define $A(x) \in \{0,1\}^n$ by setting $A(x)_i = 1$ if $x \in S_i$ and $A(x)_i = 0$ otherwise. Similarly, for each $y \in \mathcal{Y}$ define $B(y) \in \{0,1\}^n$ by setting $B(y)_i = 0$ if $y \in T_i$ and $B(y)_i = 1$ otherwise. Now define the partial monotone boolean function $f_{\mathcal{R}} : \{0,1\}^n \to \{0,1,*\}$ as follows:

$$f_{\mathcal{R}}(z) = \begin{cases} 1 & \text{if } \exists x \in \mathcal{X} : z = A(x) \\ 0 & \text{if } \exists y \in \mathcal{Y} : z = B(y) \\ * & \text{otherwise.} \end{cases}$$

By way of contradiction suppose that $f_{\mathcal{R}}$ is not well defined and let $(x, y) \in \mathcal{X} \times \mathcal{Y}$ be a pair of elements chosen so that $A(x) = B(y)$. By construction, it follows that there does not exist an $i \in [n]$ such that $(x, y) \in R_i$, contradicting the fact that $\{R_i\}$ is a rectangle covering of $\mathcal{R}$. Furthermore, it is clear from the construction that $(x, y) \in R_i$ if and only if $(A(x), A(y), i) \in \mathsf{mKW}_f$, and thus $(A(x), B(y)) \in X_i$. □

2.4 Razborov's Rank Measure

The central complexity measure studied in this thesis *rank measure*, introduced by Razborov [55], which is an elegant and powerful measure for proving lower bounds on the monotone complexity of boolean functions. Our main technical contribution is giving a clean analysis of this measure.

Definition 2.4.1. Let $\mathcal{X}, \mathcal{Y}$ be sets and let $\mathcal{R}$ be a rectangle covering of $\mathcal{X} \times \mathcal{Y}$. Let $\mathbf{F}$ be any field, and let $A : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ be any matrix over $\mathbf{F}$. The $\mathbf{F}$*-rank measure of* $\mathcal{R}$ *at* A is

$$\mu_{\mathbf{F}}(\mathcal{R}, A) = \frac{\operatorname{rank}(A)}{\max\limits_{R \in \mathcal{R}} \operatorname{rank}(A \restriction R)}.$$

The $\mathbf{F}$*-rank measure* of $\mathcal{R}$ is $\mu_{\mathbf{F}}(\mathcal{R}) = \max_A \mu_{\mathbf{F}}(\mathcal{R}, A)$, the maximum value $\mu_{\mathbf{F}}(\mathcal{R}, A)$ can take over matrices A over $\mathbf{F}$.

So, for the purpose of proving bounds on the rank measure, we ("just") need to find a matrix A over some field $\mathbf{F}$ such that the rank of A is large, but the rank of A restricted to any rectangle $R \in \mathcal{R}$ is small. In the same paper [55] in which he introduced the rank measure, Razborov proved that $\log \mu_{\mathbf{F}}(\mathsf{mKW}_f) \leq \mathsf{mD}(f)$ by exploiting the Karchmer-Wigderson connection in Theorem 2.2.2; using this he gave a simple example of a boolean function f computable in NP such that $\mu_{\mathbf{R}}(\mathsf{mKW}_f) \geq n^{\Omega(\log n)}$, implying $\Omega(\log^2 n)$ lower bounds on $\mathsf{mD}(f)$.

Gál later observed that the rank measure also lower bounds monotone span program size [25] (see Theorem 2.5.3); plugging in Razborov's lower bound from [55] immediately yielded $n^{\Omega(\log n)}$ lower bounds for monotone span programs, improving what was then the state-of-the-art. Since span programs can efficiently simulate formulas and switching networks, it follows that the rank measure is a lower bound on these models as well. We now prove that the rank measure is a lower bound on *comparator circuit size*, which is an original contribution of this thesis.

Theorem 2.4.2. *For any partial monotone boolean f and any field $\mathbf{F}$ we have*

$$\mu_{\mathbf{F}}(\mathsf{mKW}_f) \leq \mathsf{mCC}(f).$$

Proof. Define a *formal monotone complexity measure* (cf. [35]) to be any function μ mapping monotone boolean functions to nonnegative reals satisfying the following axioms, for all monotone boolean functions f, g and all coordinate functions x_i:

$$\max\{\mu(f \wedge g), \mu(f \vee g)\} \leq \mu(f) + \mu(g)$$
$$\mu(x_i) \leq 1.$$

A simple example of a formal monotone complexity measure is monotone formula size $\mathsf{mF}(f)$; furthermore, a simple induction shows that $\mathsf{mF}(f)$ is pointwise maximal amongst *all* monotone complexity measures.

If μ satisfies the stronger axiom $\mu(f) + \mu(g) \geq \mu(f \wedge g) + \mu(f \vee g)$ then μ is a *submodular complexity measure*; Razborov proved that the rank measure $\mu_{\mathbf{F}}$ is submodular.

Theorem 2.4.3 (Theorem 1, [53]). *The rank measure $\mu_{\mathbf{F}}$ is a submodular complexity measure.*

We prove that every submodular complexity measure lower bounds monotone comparator circuit size. To do this, we introduce an intermediate notion (a *multicomplexity measure*) which is easily seen to lower bound comparator circuit size. Then we show that submodular complexity measures are multicomplexity measures.

Define a function ρ to be a *multicomplexity measure* if ρ maps *sequences* of Boolean functions to nonnegative reals such that the following inequalities hold (note that all inequalities below are quantified over all sequences of boolean functions and all t, when necessary):

$$\forall i,j : \rho(f_1, f_2, \dots, f_i, \dots, f_j, \dots, f_t) \geq \rho(f_1, f_2, \dots, f_i \wedge f_j, \dots, f_i \vee f_j, \dots, f_t) \quad (2.1)$$

$$\rho(f_1, f_2, \dots, f_t) + \rho(f_{t+1}, \dots, f_m) \geq \rho(f_1, f_2, \dots, f_t, f_{t+1}, \dots, f_m) \quad (2.2)$$

$$1 \geq \rho(x_i) \quad (2.3)$$

$$\rho(f_1, f_2, \dots, f_{t-1}, f_t) \geq \rho(f_1, f_2, \dots, f_{t-1}) \quad (2.4)$$

$$\forall \pi \in \mathrm{Permutation}(t), \rho(\pi(f_1, f_2, \dots, f_t)) = \rho(f_1, f_2, \dots, f_t) \quad (2.5)$$

For any sequence of monotone boolean functions $f_1, f_2, \dots, f_t$ let $\mathsf{mCC}(f_1, f_2, \dots, f_t)$ denote the number of wires in the smallest monotone comparator circuit computing $f_1, f_2, \dots, f_t$ among its outputs.

Claim. Let $f_1, f_2, \dots, f_t$ be any sequence of monotone boolean functions and let ρ be a multicomplexity measure. Then

$$\rho(f_1, f_2, \dots, f_t) \leq \mathsf{mCC}(f_1, f_2, \dots, f_t).$$

Proof of Claim. Let t be an arbitrary positive integer, and we prove the proposition by induction on $s = \mathsf{mCC}(f_1, f_2, \dots, f_t)$. If $s = \mathsf{mCC}(f_1, \dots, f_t) = 1$ then $t = 1$ and f_1 is a variable, and so the inequality follows from (2.3). So, suppose $s > 1$ and let C be a comparator circuit witnessing $\mathsf{mCC}(f_1, \dots, f_t)$. We may assume that C has some nonzero number of comparator gates, for if C has no comparator gates then each function in $\{f_1, \dots, f_t\}$ is a variable and the proposition follows from the inductive hypothesis and repeated applications of (2.2) and (2.3). Let C' be the circuit obtained from C by removing (starting from the output) the minimum number of comparator gates $c_1, c_2, \dots, c_t$ such that C' can be partitioned into two disjoint comparator circuits C_1, C_2 with no comparator gates connecting C_1 and C_2. Clearly $s = |C_1| + |C_2|$, and let $g_1, g_2, \dots, g_i$ be the functions output by C_1 and $g_{i+1}, \dots, g_s$ the functions output by C_2. Applying the inductive hypothesis we have

$$s = |C_1| + |C_2| \geq \rho(g_1, g_2, \ldots, g_i) + \rho(g_{i+1}, \ldots, g_s) \geq \rho(g_1, \ldots, g_s),$$

where we have applied (2.2). Now, apply rule (2.1) to the pairs of wires dictated by the sequence of comparator gates $c_t, c_{t-1}, \ldots, c_1$, obtaining $s \geq \rho(g'_1, g'_2, \ldots, g'_s)$, and note that $\{f_1, \ldots, f_t\} \subseteq \{g'_1, g'_2, \ldots, g'_s\}$. Applying (2.4) finishes the proof. $\square$

Now, let $\mu(f)$ be any submodular complexity measure, and define $\rho(f_1, f_2, \ldots, f_t) = \sum_i \mu(f_i)$. We claim that ρ is a multicomplexity measure, from which $\mu_{\mathbf{F}}(f) \leq \mathsf{mCC}(f)$ immediately follows. Observe that equations (2.2), (2.3), (2.4), (2.5) easily follow from the definition of ρ and the non-negativity of μ. To see that (2.1) holds we apply submodularity:

$$\begin{aligned}\rho(f_1, f_2, \ldots, f_t) &= \mu(f_1) + \cdots + \mu(f_i) + \cdots + \mu(f_j) + \cdots + \mu(f_t) \\ &\geq \mu(f_1) + \cdots + \mu(f_i \wedge f_j) + \cdots + \mu(f_i \vee f_j) + \cdots + \mu(f_t) \\ &= \rho(f_1, f_2, \ldots, f_i \wedge f_j, \ldots f_i \vee f_j, \ldots f_t).\end{aligned}$$
$\square$

Thus, every submodular complexity measure lower bounds monotone comparator circuit size, and applying Theorem 2.4.3 completes the proof.

The next theorem summarizes the relationships between the complexity measures considered so far.

Theorem 2.4.4. *Let f be a partial monotone boolean function and let $\mathbf{F}$ be any field. Then*

$$\mu_{\mathbf{F}}(\mathsf{mKW}_f) \leq \mathsf{mSPAN}_{\mathbf{F}}(f) \leq \mathsf{mS}(f) \leq \mathsf{mF}(f),$$

and

$$\mu_{\mathbf{F}}(\mathsf{mKW}_f) \leq \mathsf{mCC}(f) \leq \mathsf{mF}(f).$$

Proof. In Chapter 1 we discussed the inequalities $\mathsf{mSPAN}_{\mathbf{F}}(f) \leq \mathsf{mS}(f) \leq \mathsf{mF}(f)$ and $\mathsf{mCC}(f) \leq \mathsf{mF}(f)$. We just proved that $\mu_{\mathbf{F}}(f) \leq \mathsf{mCC}(f)$, and in the next section (Theorem 2.5.3) we prove $\mu_{\mathbf{F}}(\mathsf{mKW}_f) \leq \mathsf{mSPAN}_{\mathbf{F}}(f)$, which was originally shown by Gál [25]. $\square$

2.5 Algebraic Tiling Number

The second complexity measure of mKW_f that we consider is the *algebraic tiling number*, introduced by Gál [25]. Just as deterministic communication complexity captures circuit depth, the algebraic tiling number captures the size of *span programs*. Following the previous section, we will define algebraic tiling in terms of arbitrary rectangle covers $\mathcal{R}$ instead of monotone boolean functions.

Recall that if A is a $\mathcal{X} \times \mathcal{Y}$ matrix and R is a combinatorial rectangle then A is *embedded* in R if A only takes non-zero values inside R, i.e. $A = A\restriction R$. The next definition[1] is due to Gál [25].

Definition 2.5.1. Let $\mathcal{X}, \mathcal{Y}$ be sets and let $\mathcal{R} = \{R_i\}_{i=1}^n$ be a rectangle covering of $\mathcal{X} \times \mathcal{Y}$. Let $\mathbf{F}$ be any field. An $\mathbf{F}$-*algebraic tiling* of $\mathcal{R}$ is a set of $\mathcal{X} \times \mathcal{Y}$ matrices $A_1, A_2, \dots, A_n$ over $\mathbf{F}$ such that $\sum_{i=1}^n A_i = \mathbb{1}$ and A_i is embedded in R_i for each i. The *size* of the algebraic tiling is $\sum_{i=1}^n \operatorname{rank}_{\mathbf{F}}(A_i)$, and the $\mathbf{F}$-*algebraic tiling number* of $\mathcal{R}$ is the minimum size $\chi_{\mathbf{F}}(\mathcal{R})$ of any $\mathbf{F}$-algebraic tiling of $\mathcal{R}$.

Theorem 2.5.2 (Theorem 3.4 in [25])**.** *For any partial monotone boolean function f and any field $\mathbf{F}$, $\mathsf{mSPAN}_{\mathbf{F}}(f) = \chi_{\mathbf{F}}(\mathsf{mKW}_f)$.*

With the definition of algebraic tiling in hand, it is easy to see that Razborov's rank measure is a lower bound on monotone span program size. As usual, we will state it in terms of rectangle coverings.

Theorem 2.5.3 (Lemma 3.2 in [25])**.** *Let $\mathcal{X}, \mathcal{Y}$ be sets and let $\mathcal{R}$ be a rectangle covering of $\mathcal{X} \times \mathcal{Y}$. For any field $\mathbf{F}$ we have $\mu_{\mathbf{F}}(\mathcal{R}) \leq \chi_{\mathbf{F}}(\mathcal{R})$.*

Proof. Write $\mathcal{R} = \{R_i\}_{i=1}^n$. Let A be any $\mathcal{X} \times \mathcal{Y}$ matrix and let $B_1, B_2, \dots, B_n$ be any algebraic tiling of $\mathcal{R}$. We show $\mu_{\mathbf{F}}(\mathcal{R}, A) \leq \sum_{i=1}^n \operatorname{rank}(B_i)$, which implies the theorem.

Let $\mathbb{1}$ denote the $\mathcal{X} \times \mathcal{Y}$ all-1s matrix, and for two matrices X, Y let $X * Y$ denote their Hadamard (i.e. entrywise) product. The definition of an algebraic tiling states that $\mathbb{1} = \sum_{i=1}^n B_i$. Since B_i is embedded in R_i we have $B_i = B_i\restriction R_i = B_i * (\mathbb{1}\restriction R_i)$; using this fact and taking Hadamard products with A yields

$$A = \sum_{i=1}^n A * B_i * (\mathbb{1}\restriction R_i) = \sum_{i=1}^n (A * (\mathbb{1}\restriction R_i)) * B_i = \sum_{i=1}^n (A\restriction R_i) * B_i.$$

Taking ranks, and noting that rank is sub-additive and sub-multiplicative with respect to Hadamard products, we get

$$\operatorname{rank}(A) \leq \sum_{i=1}^n \operatorname{rank}(A\restriction R_i) \operatorname{rank}(B_i) \leq \max_{i \in [n]} \operatorname{rank}(A\restriction R_i) \sum_{i=1}^n \operatorname{rank}(B_i).$$

[1]Our definition is slightly modified from the one in [25]. There, an algebraic tiling consists of a set of rank-1 matrices $A_1, ..., A_t$ such that each A_i is embedded in some $R \in \mathcal{R}$, and the size of the tiling is t. Our definition is easily seen to be equivalent by taking a rank-1 decomposition of the matrices in each rectangle.

Dividing through by $\max_{i\in[n]} \mathrm{rank}(A{\restriction}R_i)$ yields the theorem. $\square$

Chapter 3

Pattern Matrices and the Rank Measure

Let us now briefly stop and take stock. In the previous chapter, we introduced the *Karchmer-Wigderson games*, which provide a framework in the setting of communication complexity for studying complexity lower bounds in these models. We discussed two complexity measures of Karchmer-Wigderson games: first, the *rank measure*, due to Razborov, provides lower bounds on nearly every circuit model that we have considered; second, the *algebraic tiling number*, due to Gál, characterizes the size of span programs. The main technical goal is to analyze the rank measure and the algebraic tiling number; this task will be made much easier (conceptually, at least) by considering arbitrary rectangle covers $\mathcal{R}$ instead of the Karchmer-Wigderson games associated with fixed monotone boolean functions.

The main difficulty in analyzing the rank measure $\mu_{\mathbf{F}}(\mathcal{R})$ is obvious: how do we define a matrix A such that $\mathrm{rank}(A)$ is large while $\mathrm{rank}(A\restriction R)$ is small for *every* rectangle $R \in \mathcal{R}$? Our approach is to restrict A to be a special type of matrix generated by a multilinear polynomial called a *pattern matrix*. In this chapter we define pattern matrices and show how we can exploit their structure to bound the rank measure. Then, as a warmup, we will prove a simple version of our main theorem for the real rank measure $\mu_{\mathbb{R}}(\mathcal{R})$.

3.1 Multilinear Polynomials and Pattern Matrices

Recall a polynomial $p \in \mathbf{F}[z_1, z_2, \dots z_n]$ is *multilinear* if the maximum degree of each variable z_i in p is 1. If p is multilinear, it follows that all terms in p are products of variables $\prod_{i\in S} z_i$ for some $S \subseteq [n]$, and thus it has at most 2^n distinct monomials. Given a multilinear polynomial p, we will borrow notation from Fourier analysis and let $\hat{p}(S) \in \mathbf{F}$ denote the coefficient of the monomial $\prod_{i\in S} z_i$ in p, allowing us to write

$$p = \sum_{S\subseteq[n]} \hat{p}(S) \prod_{i\in S} z_i.$$

Finally, if $\pi : [n] \to \mathbf{F} \cup \{*\}$ is a partial restriction of the variables of p, then let $p\restriction\pi$ denote the polynomial over the unrestricted variables of π obtained from p in the natural way.

Definition 3.1.1. Let $\mathbf{F}$ be a field and let $p \in \mathbf{F}[z_1, z_2, \ldots, z_n]$ be a multilinear polynomial over $\mathbf{F}$. Let $\mathcal{X}, \mathcal{Y}$ be sets and let $g : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ be any function. The *pattern matrix* obtained from p and g is the matrix $p \circ g^n : \mathcal{X}^n \times \mathcal{Y}^n \to \mathbf{F}$ defined by composing p with n independent copies of g. Formally, for any $(x, y) \in \mathcal{X}^n \times \mathcal{Y}^n$ let

$$(p \circ g^n)(x, y) = \sum_{S \subseteq [n]} \hat{p}(S) \prod_{i \in S} g(x_i, y_i).$$

Following the literature, we will refer to the function g used in the construction above as a *gadget*.

Pattern matrices were introduced by Sherstov [61] in the special case where p is a real multilinear polynomial and g was chosen to be a particular gadget g_λ; using these pattern matrices he was able to prove strong lower bounds on quantum communication complexity. A particular result from [61] that is useful to us is a characterization of the rank of real pattern matrices generated by g_λ.

Theorem 3.1.2 (Corollary of Theorem 4.3 in [61])**.** *Let $p \in \mathbb{R}[z_1, z_2, \ldots, z_n]$ be a real multilinear polynomial. For every positive integer λ there exists a gadget $g_\lambda : ([\lambda] \times \{-1, 1\}) \times \{-1, 1\}^\lambda \to \{-1, 1\}$ such that*

$$\operatorname{rank}_{\mathbb{R}}(p \circ g^n) = \sum_{S : \hat{p}(S) \neq 0} \lambda^{|S|}.$$

In a later chapter (cf. Section 5.1) we prove a strong generalization of this theorem that works for arbitrary fields and for any g satisfying a certain generic property.

Our plan for this chapter is to use pattern matrices to lower bound $\mu_{\mathbb{R}}(\mathcal{R})$ for some special rectangle covering $\mathcal{R}$ via Theorem 3.1.2. By definition, the real rank measure of $\mathcal{R}$ at a pattern matrix $p \circ g^n$ is

$$\mu_{\mathbb{R}}(\mathcal{R}, p \circ g^n) = \frac{\operatorname{rank}_{\mathbb{R}}(p \circ g^n)}{\max\limits_{R \in \mathcal{R}} \operatorname{rank}_{\mathbb{R}}((p \circ g^n)\restriction R)}.$$

Using Theorem 3.1.2 we can calculate the numerator, but how do we bound $\operatorname{rank}_{\mathbb{R}}((p \circ g^n)\restriction R)$ for any $R \in \mathcal{R}$? Our approach is simple: we will *define* the rectangle covering $\mathcal{R}$ so that $(p \circ g^n)\restriction R$ is *also* a pattern matrix for every rectangle R. This will allow us to apply Theorem 3.1.2 again to calculate the denominator!

How can we define a rectangle cover to satisfy this property? In a word, we will choose the rectangle cover $\mathcal{R}$ so that the *rectangles in the cover* correspond to *restrictions of the* *polynomial*. This is best illustrated by a simple example. Let $p \in \mathbb{R}[z_1, z_2, \ldots, z_n]$, and let $t \leq n$. For each $i \in [t]$, fix $x_i^* \in \mathcal{X}$ and $y_i^* \in \mathcal{Y}$, and consider the combinatorial rectangle

$$R = \{x \in \mathcal{X}^n \mid \forall i \leq t : x_i = x_i^*\} \times \{y \in \mathcal{Y}^n \mid \forall i \leq t : y_i = y_i^*\}$$

by restricting the first t coordinates of x, y to x_i^*, y_i^*, respectively. Finally, let

$$\pi = (g(x_1^*, y_1^*), g(x_2^*, y_2^*), \ldots, g(x_t^*, y_t^*))$$

be the tuple of values obtained by evaluating g at the (x_i^*, y_i^*) pairs. By definition, for all $(x, y) \in \mathcal{R}$ we have $g^n(x, y) = \pi g(x_{t+1}, y_{t+1}) \cdots g(x_n, y_n)$; in words, for every assignment (x, y) in the rectangle the first t inputs always evaluate to π. This means we can write $(p \circ g^n)\restriction R = (p\restriction\pi) \circ g^{n-t}$; the second expression is clearly a pattern matrix of the restricted polynomial $p\restriction\pi$ on a subset of the input variables to p. (Our actual construction will be more involved, but this is the key idea.)

In the next section, we give a generic way to construct such special rectangle covers using *unsatisfiable CNF search problems*. We then prove the main theorem of this chapter, which reduces the problem of lower-bounding the rank measure to bounding a different quantity on polynomials that we call the *algebraic gap complexity* (albeit, only for real pattern matrices due to the use of Theorem 3.1.2). Peeking forward: in the next chapter we will see that the algebraic gap complexity is *exactly* the same as the classical *Nullstellensatz degree*; ultimately this will allow us to transfer Nullstellensatz degree lower bounds (which are abundant in the literature, e.g [6, 13, 15, 17]) to lower bounds on the rank measure.

3.2 CNF-Search Problems and Canonical Rectangle Covers

In this section we show how to construct the "special" rectangle covers for which we prove rank measure lower bounds. These covers have been implicitly used in other works, e.g. [28, 45, 49]. The starting point is the following search problem associated with unsatisfiable CNF formulas.

Definition 3.2.1. Let $\mathcal{C} = \bigwedge_{i=1}^m C_i$ be an unsatisfiable boolean formula on n variables in conjunctive normal form (CNF). The *CNF-Search* problem for $\mathcal{C}$, denoted $\mathsf{Search}(\mathcal{C})$, is defined as follows: given an assignment $x \in \{0,1\}^n$ to the variables of $\mathcal{C}$, output any clause C_i that is falsified by x.

Since $\mathcal{C}$ is unsatisfiable the search problem $\mathsf{Search}(\mathcal{C})$ is *total*: for every input x some clause will be falsified. Using $\mathsf{Search}(\mathcal{C})$ and any gadget $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ we can construct a rectangle cover $\mathcal{R}$ of $\mathcal{X}^n \times \mathcal{Y}^n$ by exploiting this totality. We record this construction in the next definition.

Definition 3.2.2. Let $\mathcal{C}$ be an unsatisfiable CNF on n boolean variables and let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be any gadget. The *canonical rectangle cover* $\mathcal{R}_{\mathcal{C},g}$ of $\mathcal{C}$ and g is defined as follows. For every clause $C \in \mathcal{C}$ and every assignment $\alpha \in \mathcal{X}^{\mathsf{vars}(C)}$ define the rectangle

$$\begin{aligned} R_{C,\alpha} &= \left\{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n \mid x_{\mathsf{vars}(C)} = \alpha \text{ and } g^n(x,y) \text{ falsifies } C\right\} \\ &= \left\{x \in \mathcal{X}^n \mid x_{\mathsf{vars}(C)} = \alpha\right\} \times \left\{y \in \mathcal{Y}^n \mid g^{\mathsf{vars}(C)}(\alpha, y_{\mathsf{vars}(C)}) \text{ falsifies } C\right\} \end{aligned}$$

and set $\mathcal{R}_{\mathcal{C},g} = \left\{R_{C,\alpha} \mid C \in \mathcal{C}, \alpha \in \mathcal{X}^{\mathsf{vars}(C)}\right\}$.

The definition of the canonical rectangle cover is rather technical, so let us pull it apart in steps. Given the search problem $\mathsf{Search}(\mathcal{C})$, one can consider the following natural search problem $\mathsf{Search}(\mathcal{C}, g) \subseteq \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{C}$ obtained by replacing each variable z_i in $\mathsf{Search}(\mathcal{C})$ with $g(x_i, y_i)$ for fresh variables (x_i, y_i): then, given $(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n$ the goal is to evaluate $\mathsf{Search}(\mathcal{C})$ on the string $z = g^n(x,y)$. This search problem is total since $\mathsf{Search}(\mathcal{C})$ is, and so we can try and exploit this totality to get a rectangle covering of $\mathcal{X}^n \times \mathcal{Y}^n$. A natural first attempt is to consider, for each clause $C \in \mathcal{C}$, the set

$$R_C = \left\{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n \mid g^{\mathsf{vars}(C)}(x_{\mathsf{vars}(C)}, y_{\mathsf{vars}(C)}) \text{ falsifies } C\right\}.$$

Since $\mathcal{C}$ is unsatisfiable it follows that $\{R_C\}$ covers all inputs in $\mathcal{X}^n \times \mathcal{Y}^n$; however R_C may not in general be a combinatorial rectangle. The definition of the canonical cover fixes this by "breaking up" each of the sets R_C into rectangles by ranging over all fixed assignments to $x_{\mathsf{vars}(C)}$. This is roughly depicted in Figure 3.1.

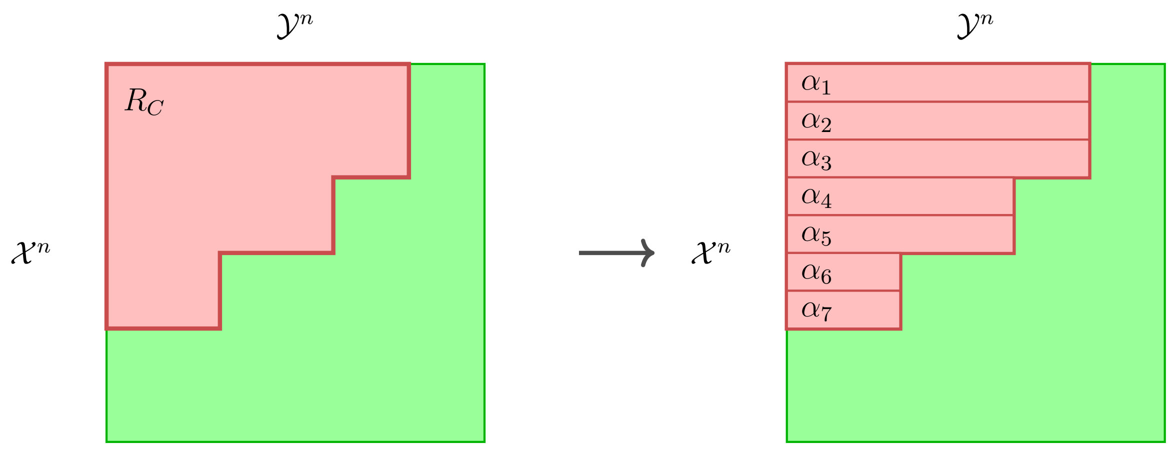

Figure 3.1: Construction of $\mathcal{R}_{\mathcal{C},g}$.

Each set R_C is thus replaced with at most $|\mathcal{X}|^k$ rectangles, where k is the width of the widest clause in $\mathcal{C}$. Recalling how we transform rectangle covers into monotone boolean functions (cf. Proposition 2.3.3), the size of the cover is exactly the number of input variables of the resulting boolean function — so, we will able to afford the blow-up as long as $|\mathcal{X}|^k$ is polynomially bounded. We record this bound on the size of the canonical cover in the next proposition.

Proposition 3.2.3. *Let* $\mathbf{F}$ *be any field. Let* $\mathcal{C}$ *be a width-*k *unsatisfiable CNF and let* $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. *Then* $|\mathcal{R}_{\mathcal{C},g}| \leq |\mathcal{C}||\mathcal{X}|^k$.

Proof. The bound on the size of $\mathcal{R}_{\mathcal{C},g}$ follows immediately from the definition since there are $|\mathcal{C}|$ clauses and each clause has at most k variables. □

3.3 Lifting Algebraic Gaps to Razborov's Rank Measure (Real Case)

In this section we bring the results of this chapter together and analyze $\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g})$. To be precise, we will *not* be directly lower bounding $\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g})$ with the technology we have introduced; rather, the main theorem *reduces* the problem of lower bounding $\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g})$ to lower bounding a quantity on unsatisfiable CNFs that we call the *algebraic gap complexity*, introduced next. (We remark that the next definition is a special case of a more general definition given in the next chapter (cf. Section 4.2).)

Definition 3.3.1. Let $\mathcal{C}$ be an unsatisfiable CNF on n variables, and for each clause C define the restriction $\pi_C : [n] \to \{-1, 1, *\}$ to be a $\{-1, 1\}$ restriction to the variables of $\mathcal{C}$ by

$$\pi_C(z_i) := \begin{cases} -1 & z_i \text{ is true in any falsifying assignment of } C \\ 1 & z_i \text{ is false in any falsifying assignment of } C \\ * & \text{otherwise.} \end{cases}$$

The *real algebraic gap complexity* of $\mathcal{C}$ is the largest integer $\mathsf{gap}_{\mathbb{R}}(\mathcal{C}) \in \mathbb{Z}$ for which there exists a real multilinear polynomial $p \in \mathbb{R}[z_1, z_2, \ldots, z_n]$ such that

$$\deg(p) = n \quad \text{and} \quad \forall C \in \mathcal{C} : \deg(p\restriction\pi_C) \leq n - \mathsf{gap}_{\mathbb{R}}(\mathcal{C}).$$

We note that we only use $\{-1, 1\}$ restrictions here since we will be using Sherstov's gadget g_λ from Theorem 3.1.2 which outputs $\{-1, 1\}$ values; we can convert a $\{-1, 1\}$ assignment to a $\{0, 1\}$ assignment using the mapping $z_i \mapsto (1 - z_i)/2$.

There is an obvious analogy between the algebraic gap complexity and the rank measure: in the rank measure we seek to find a high-rank matrix A such that $\mathrm{rank}(A\restriction R)$ is small for every rectangle R; similarly, in the algebraic gap complexity we seek to find a multilinear polynomial p such that $\deg(p)$ is large and $\deg(p{\upharpoonright}\pi_C)$ is small for every restriction π_C. The main theorem of this chapter is a reduction from the real rank measure to the real algebraic gap complexity.

Theorem 3.3.2. *Let n, k be positive integers and let $\mathcal{C}$ be an unsatisfiable k-CNF on n variables. There exists a gadget $g : \mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$ with $|\mathcal{X}| = 2n^2$ such that*

$$\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g}) \geq \Omega(n^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})}).$$

Proof. Choose the gadget $g = g_\lambda$ from Theorem 3.1.2 where $\lambda = n^2$. Let $p \in \mathbb{R}[z_1, z_2, \ldots, z_n]$ be the polynomial witnessing the algebraic gap complexity $\mathsf{gap}_{\mathbb{R}}(\mathcal{C})$, and let $A = p \circ g^n$ be the pattern matrix obtained by composing p and g. We prove

$$\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g}, A) = \frac{\operatorname{rank}_{\mathbb{R}}(A)}{\max\limits_{R \in \mathcal{R}_{\mathcal{C},g}} \operatorname{rank}_{\mathbb{R}}(A{\upharpoonright}R)} \geq \Omega(n^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})}).$$

Applying Theorem 3.1.2 we have

$$\operatorname{rank}_{\mathbb{R}}(A) = \sum_{S : \hat{p}(S) \neq 0} \lambda^{|S|} = \sum_{S : \hat{p}(S) \neq 0} n^{2|S|} \geq n^{2n}$$

since $\deg p = n$ by the definition of $\mathsf{gap}_{\mathbb{R}}(\mathcal{C})$. For the denominator, let $R_{C,\alpha}$ be an arbitrary rectangle from the cover $\mathcal{R}_{\mathcal{C},g}$. We claim that

$$\operatorname{rank}_{\mathbb{R}}(A{\upharpoonright}R_{C,\alpha}) = \sum_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} n^{2|S|}, \tag{3.1}$$

To see Equation 3.1, we claim that the matrix $A{\upharpoonright}R_{C,\alpha}$ is column-equivalent to the block matrix

$$[(p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}, (p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}, \ldots, (p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}]$$

for some number of copies of the matrix $(p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}$. Equation 3.1 immediately follows from the claim as

$$\operatorname{rank}_{\mathbb{R}}(A{\upharpoonright}R_{C,\alpha}) = \operatorname{rank}_{\mathbb{R}}((p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}) = \sum_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} n^{2|S|}$$

by Theorem 3.1.2. So let us prove the claim.

By the definition, for all $(x, y) \in R_{C,\alpha}$ we have that $g^{\mathsf{vars}(C)}(x_{\mathsf{vars}(C)}, y_{\mathsf{vars}(C)}) = \pi$ and $x_{\mathsf{vars}(\pi)} = \alpha$. Let us first fix any assignment β to $y_{\mathsf{vars}(C)}$ so that $g^{\mathsf{vars}(C)}(\alpha, \beta) = \pi$. Then for all $(x, y) \in R_{C,\alpha}$ such that $y_{\mathsf{vars}(C)} = \beta$ we have

$$\begin{aligned} g^n(x,y) &= g^{\mathsf{vars}(C)}(\alpha,\beta) g^{[n]\setminus\mathsf{vars}(C)}(x_{[n]\setminus\mathsf{vars}(C)}, y_{[n]\setminus\mathsf{vars}(C)}) \\ &= \pi g^{[n]\setminus\mathsf{vars}(C)}(x_{[n]\setminus\mathsf{vars}(C)}, y_{[n]\setminus\mathsf{vars}(C)}), \end{aligned}$$

and so ranging $x_{[n]\setminus\mathsf{vars}(C)}, y_{[n]\setminus\mathsf{vars}(C)}$ over all values yields the matrix $(p{\restriction}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}$. Then, ranging $y_{\mathsf{vars}(C)}$ over all β such that $g^{\mathsf{vars}(C)}(\alpha, \beta) = \pi$ yields the claim and Equation 3.1.

Applying Equation 3.1, we have

$$\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g}, A) = \frac{\sum_{S:\hat{p}(S)\neq 0} n^{2|S|}}{\max_{R_{C,\alpha}\in\mathcal{R}_{\mathcal{C},g}} \sum_{S:\widehat{p{\restriction}\pi}(S)\neq 0} n^{2|S|}} \geq \frac{n^{2n}}{\max_{R_{C,\alpha}\in\mathcal{R}(\mathcal{C},g)} \sum_{S:\widehat{p{\restriction}\pi}(S)\neq 0} n^{2|S|}}$$

where the inequality follows since $\deg p = n$. Since p witnesses the algebraic gap of $\mathcal{C}$, we have that $\deg p{\restriction}\pi \leq n - \mathsf{gap}_{\mathbb{R}}(\mathcal{C})$ for all π. We may clearly assume that $\hat{p}(S) = 0$ when $|S| < n - \mathsf{gap}_{\mathbb{R}}(\mathcal{C})$, so, for any π:

$$\begin{aligned} \sum_{S:\widehat{p{\restriction}\pi}(S)\neq 0} n^{2|S|} &\leq \sum_{i=0}^{k} \binom{n}{\mathsf{gap}_{\mathbb{R}}(\mathcal{C}) - i} n^{2(n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i)} \\ &\leq \sum_{i=0}^{k} \left(\frac{en}{\mathsf{gap}_{\mathbb{R}}(\mathcal{C}) - i}\right)^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i} n^{2(n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i)} \\ &= \sum_{i=0}^{k} \left(\frac{e}{\mathsf{gap}_{\mathbb{R}}(\mathcal{C}) - i}\right)^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i} n^{2n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i} \\ &\leq n^{2n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})} \sum_{i=0}^{k} \left(\frac{e}{\mathsf{gap}_{\mathbb{R}}(\mathcal{C}) - i}\right)^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})-i} \leq 6n^{2n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})} \end{aligned}$$

since $e + (e/2)^2 + (e/3)^3 + \cdots \leq 6$. Putting it all together, we get

$$\mu_{\mathbb{R}}(\mathcal{R}_{\mathcal{C},g}, A) \geq \frac{n^{2n}}{6n^{2n-\mathsf{gap}_{\mathbb{R}}(\mathcal{C})}} = cn^{\mathsf{gap}_{\mathbb{R}}(\mathcal{C})}$$

where $c = 1/6$, proving the theorem. □

Now it remains to prove lower bounds on $\mathsf{gap}_{\mathbb{R}}(\mathcal{C})$ for some unsatisfiable CNF $\mathcal{C}$ in order to obtain lower bounds on the rank measure. We will attack this in the next chapter.

Chapter 4

Algebraic Gaps and Nullstellensatz

Now that we have seen a "warmup" version of our lifting theorem in Theorem 3.3.2, in this chapter we investigate the algebraic gap complexity. Ultimately, we will show that algebraic gaps are exactly the same as the *Nullstellensatz degree* in proof complexity. This will allow us to appeal to the broad set of Nullstellensatz lower bounds in the literature to obtain lower bounds on the rank measure.

4.1 Nullstellensatz Refutations

We begin by reviewing the Nullstellensatz proof system [6].

Definition 4.1.1. Let $\mathbf{F}$ be a field, and let $\mathcal{P} = \{p_1 = 0, p_2 = 0, \ldots, p_m = 0\}$ be an unsatisfiable system of polynomial equations in $\mathbf{F}[z_1, z_2, \ldots, z_n]$. A *Nullstellensatz refutation* of $\mathcal{P}$ is a sequence of polynomials $q_1, q_2, \ldots, q_m \in \mathbf{F}[z_1, z_2, \ldots, z_n]$ such that $\sum_{i=1}^m p_i q_i = 1$ where the equality is syntactic. The *degree* of the refutation is $\max_i \deg(p_i q_i)$; the *Nullstellensatz degree* of $\mathcal{P}$, denoted $\mathsf{NS}_{\mathbf{F}}(\mathcal{P})$, is the minimum degree of any Nullstellensatz refutation of $\mathcal{P}$.

It is fruitful to compare this definition with Definition 2.5.1: Nullstellensatz degree is the analogue of the algebraic tiling number for polynomials.

The name "Nullstellensatz" comes from *Hilbert's Nullstellensatz*, which is a central theorem in modern algebraic geometry that links the zeros of a system of polynomials (*varieties*) to polynomials that are derivable from the system (*ideals*). In computational complexity (and, more specifically, propositional proof complexity), the Nullstellensatz proof system is the prototypical example of an *algebraic proof system* for refuting propositional contradictions [6]. Of course, in order to refute propositional contradictions we first need to discuss how to *encode* propositional formulas into systems of polynomials; we do this next.

Let $\mathcal{C} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF formula over boolean variables $z_1, z_2, \ldots, z_n$. We introduce a standard encoding of each clause C_i as a polynomial equation.

If C is a clause then let C^+ denote the set of variables occurring positively in C and C^- denote the set of variables occurring negatively in C; with this notation we can write $C = \bigvee_{z \in C^+} z \vee \bigvee_{z \in C^-} \overline{z}$. From C define the polynomial

$$\mathcal{E}(C) \equiv \prod_{z \in C^+} (1 - z) \prod_{z \in C^-} z,$$

and observe that $\mathcal{E}(C) = 0$ is satisfied (over $0/1$ assignments to z_i) if and only if the corresponding assignment satisfies C. We abuse notation and let $\mathcal{E}(\mathcal{C}) = \{\mathcal{E}(C) \mid C \in \mathcal{C}\} \cup \{z_i^2 - z_i\}_{i \in [m]}$, and note that the second set of polynomial equations restricts the z_i inputs to $\{0, 1\}$ values.

Definition 4.1.2. Let $\mathcal{C}$ be an unsatisfiable CNF formula and let $\mathbf{F}$ be a field. The $\mathbf{F}$-*Nullstellensatz degree* of $\mathcal{C}$, denoted $\mathsf{NS}_{\mathbf{F}}(\mathcal{C})$, is the Nullstellensatz degree of refuting $\mathcal{E}(\mathcal{C})$.

How do we know that a Nullstellensatz refutation always exists? One can deduce this from Hilbert's Nullstellensatz, but, for our purposes, we will use a simple version of it proved by Buss et al. [16].

Theorem 4.1.3 (Theorem 5.2 in [16])**.** *Let* $\mathbf{F}$ *be any field and let* $\mathcal{P}$ *be any system of polynomial equations over* $\mathbf{F}[z_1, \ldots, z_n]$ *with no* $\{0, 1\}$ *solutions. Then there exists a Nullstellensatz refutation of* $\mathcal{P} \cup \{z_i^2 - z_i = 0\}_{i \in [n]}$.

In proof complexity the Nullstellensatz proof system has been extensively studied (e.g. [6, 14–17]) and we now have very strong lower bounds on the Nullstellensatz degree for a variety of unsatisfiable systems of polynomial equations.

4.2 Algebraic Gaps: A General Definition

We will now give a general definition of algebraic gaps over arbitrary fields. For technical convenience, we first introduce the notion of a *certificate* of an unsatisfiable CNF formula.

Definition 4.2.1. Let $\mathcal{C}$ be an unsatisfiable boolean formula on n variables in conjunctive normal form (CNF), and let C be a clause in $\mathcal{C}$. The *certificate* of C, denoted $\mathsf{Cert}(C)$, is the partial assignment $\pi : [n] \to \{0, 1, *\}$ which falsifies C and sets the maximal number of variables to $*$s. Let $\mathsf{Cert}(\mathcal{C})$ denote the set of all certificates of clauses of $\mathcal{C}$.

Say that an assignment $z \in \{0, 1\}^n$ *agrees* with a certificate $\pi \in \mathsf{Cert}(\mathcal{C})$ if $\pi(i) = z_i$ for each i assigned to a $\{0, 1\}$ value by π. Since the CNF formula $\mathcal{C}$ is unsatisfiable, it follows that every assignment in $z \in \{0, 1\}^n$ agrees with some $\{0, 1\}$-certificate of $\mathcal{C}$ (in this sense, certificates are quite similar to rectangle covers).

Definition 4.2.2. Let $\mathbf{F}$ be a field. Let $\mathcal{C}$ be an unsatisfiable CNF on n variables. The $\mathbf{F}$-*algebraic gap complexity* of $\mathcal{C}$ is the maximum positive integer $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) \in \mathbb{N}$ for which there exists a multilinear polynomial $p \in \mathbf{F}[z_1, z_2, \dots, z_n]$ such that

$$\deg(p) = n \quad \text{and} \quad \forall \pi \in \mathsf{Cert}(\mathcal{C}) : \deg(p \restriction \pi) \leq n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C}).$$

The above definition of the gap complexity generalizes the definition of the *real* algebraic gap complexity from Definition 3.3.1 to all fields — observe that the notion of a "certificate" captures the partial assignments used in the earlier definition. For a simple example, consider the following unsatisfiable CNF formula:

$$\mathcal{C} = \overline{z}_1 \wedge \overline{z}_2 \wedge \cdots \wedge \overline{z}_n \wedge \left(\bigvee_{i \in [n]} z_i \right)$$

which asserts that every z_i must be false and that some z_i must be true. For each of the first n clauses the corresponding certificate sets $z_i = 1$ and does not set any other variable, and the certificate for the final clause sets all variables to 0. What is the *real* algebraic gap complexity $\mathsf{gap}_{\mathbf{R}}$ of $\mathcal{C}$? An auspicious choice is a polynomial encoding of the OR function OR_n, which outputs 1 if one of its inputs is 1. It is well known that $\deg_{\mathbb{R}}(OR_n) = n$; furthermore, if we restrict any input variable to 1 or all inputs to 0 then OR_n simplifies to a constant, and thus its degree is 0. This implies that $\mathsf{gap}_{\mathbb{R}}(OR_n) = n$ — the upper bound is trivial, and the lower bound follows by the previous argument.

As alluded to earlier, one can think of algebraic gaps as a "polynomial analogue" of Razborov's rank measure $\mu_{\mathbf{F}}$. If we similarly think of Nullstellensatz refutations as a "polynomial analogue" of the algebraic tiling number $\chi_{\mathbf{F}}$, then Theorem 2.5.3 (which states $\mu_{\mathbf{F}}(\mathcal{R}) \leq \chi_{\mathbf{F}}(\mathcal{R})$) suggests that $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) \leq \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$. We now prove that this is indeed the case. This is obtained by mimicking the proof of Theorem 2.5.3, where rank is replaced by degree and Hadamard products are replaced with polynomial multiplication.

Theorem 4.2.3. *For any unsatisfiable CNF formula $\mathcal{C}$ and any field $\mathbf{F}$, $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) \leq \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$.*

Proof. Let n be the number of variables of $\mathcal{C}$ and let $p \in \mathbf{F}[z_1, \dots, z_n]$ be the multilinear polynomial witnessing the algebraic gap of $\mathcal{C}$. Given a polynomial $q \in \mathbf{F}[z_1, \dots, z_n]$ let $\deg_{\text{mult}}(q)$ denote the *multilinear degree* of q (i.e. the degree of the polynomial resulting from q by replacing every term of the form z_i^k with z_i in q). For any polynomial q it clearly holds that $\deg_{\text{mult}}(q) \leq \deg(q)$, with equality holding when q is multilinear.

Write $\mathcal{C} = \bigwedge_{i=1}^{m} C_i$ for clauses C_i, and consider any minimum-degree Nullstellensatz refutation of $\mathcal{E}(\mathcal{C})$, which we write as

$$1 = \sum_{i=1}^{m} \mathcal{E}(C_i)q_i + \sum_{j=1}^{n} r_j(z_j^2 - z_j)$$

for polynomials $q_i, r_j \in \mathbf{F}[z_1, \dots, z_n]$. Multiplying through by p and taking multilinear degrees of both sides we get

$$\begin{aligned}\deg_{\text{mult}}(p) &= \max\left\{\deg_{\text{mult}}(p\mathcal{E}(C_i)q_i)\right\}_{i\in[m]} \cup \left\{\deg_{\text{mult}}(pr_j(z_j^2 - z_j))\right\}_{j\in[n]} \\ &= \max\left\{\deg_{\text{mult}}(p\mathcal{E}(C_i)q_i)\right\}_{i\in[m]}\end{aligned}$$

where the second equality follows since $\deg_{\text{mult}}(pr_j(z_j^2 - z_j)) = \deg_{\text{mult}}(0) = 0$.

Now, for every $\{0,1\}$-assignment z^*, either $\mathcal{E}(C_i)$ evaluates to 0 (in which case the clause is satisfied) or $\mathcal{E}(C_i)$ evaluates to 1, if z^* is consistent with the certificate $\pi_i = \mathsf{Cert}(C_i)$. It follows that for all $z^* \in \{0,1\}^n$ and all i we have $(p\mathcal{E}(C_i)q_i)(z^*) = ((p\restriction\pi_i)\mathcal{E}(C_i)q_i)(z^*)$, and thus $\deg_{\text{mult}}(p\mathcal{E}(C_i)q_i) = \deg_{\text{mult}}((p\restriction\pi_i)\mathcal{E}(C_i)q_i))$. Extracting the maximum over certificates, we obtain

$$\deg_{\text{mult}}(p) \le \max_{\pi\in\mathsf{Cert}(\mathcal{C})} \deg_{\text{mult}}(p\restriction\pi) + \max\left\{\deg_{\text{mult}}(\mathcal{E}(C_i)q_i)\right\}_{i\in[m]}.$$

Since $\deg_{\text{mult}}(p) = \deg(p)$ and $\deg_{\text{mult}}(\mathcal{E}(C_i)q_i) \le \deg(\mathcal{E}(C_i)q_i))$, we have

$$\begin{aligned}\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) &= \deg(p) - \max_{\pi\in\mathsf{Cert}(\mathcal{C})} \deg(p\restriction\pi) \\ &= \deg_{\text{mult}}(p) - \max_{\pi\in\mathsf{Cert}(\mathcal{C})} \deg_{\text{mult}}(p\restriction\pi) \\ &\le \max\left\{\deg_{\text{mult}}(\mathcal{E}(C_i)q_i)\right\}_{i\in[m]} \\ &\le \max\left\{\deg(\mathcal{E}(C_i)q_i)\right\}_{i\in[m]} \le \mathsf{NS}_{\mathbf{F}}(\mathcal{C}).\end{aligned}$$

□

In the remainder of the chapter we show that algebraic gaps and Nullstellensatz degree are actually the *same*. This is quite surprising, as it seems unlikely that the rank measure is the same as the algebraic tiling number.

4.3 Algebraic Gaps = Nullstellensatz (Characteristic 2 Case)

Our goal in the remainder of the chapter is to prove an upper bound version of Theorem 4.2.3, showing that algebraic gaps and Nullstellensatz degree are the same measure.

Theorem 4.3.1. *For any unsatisfiable CNF $\mathcal{C}$ and any field $\mathbf{F}$, $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$.*

In this section we study the characteristic 2 case, as it is a bit simpler and illustrates the main idea. We crucially use the following dual characterization of Nullstellensatz degree by *d-designs* [14, 17].

Definition 4.3.2. Let $\mathbf{F}$ be a field, and let $\mathcal{P}$ be an unsatisfiable system of polynomial equations over $\mathbf{F}[z_1, z_2, \ldots, z_n]$. A d-design for $\mathcal{P}$ is a linear functional D on the space of polynomials satisfying the following axioms:

1. $D(1) = 1$.
2. For all $p \in \mathcal{P}$ and all polynomials q such that $\deg(pq) \leq d$, we have $D(pq) = 0$.

It is known (see, for example, [14]) that the system $\mathcal{P}$ does not have a Nullstellensatz refutation of degree d if and only if it has a d-design, and thus every unsatisfiable system of polynomial equations $\mathcal{P}$ has an $(\mathsf{NS}(\mathcal{P}) - 1)$-design.

Before we begin, we will need the following easy lemma regarding the *dual* of a CNF. Let $\mathcal{C}$ be an unsatisfiable CNF. For any clause $C \in \mathcal{C}$ let $C^\dagger$ denote the clause obtained by negating every literal in C (so, z is replaced with $\neg z$ and $\neg z$ is replaced with z). Let $\mathcal{C}^\dagger$ be the CNF obtained from $\mathcal{C}$ by replacing each clause in $\mathcal{C}$ with its dual, and note that $\mathcal{C}^\dagger$ is unsatisfiable if and only if $\mathcal{C}$ is unsatisfiable.

Lemma 4.3.3. *For any field* $\mathbf{F}$, $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbf{F}}(\mathcal{C}^\dagger)$.

Proof. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ be an unsatisfiable CNF over variables $z_1, z_2, \ldots, z_n$. We prove $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}(\mathcal{C})) = \mathsf{NS}_{\mathbf{F}}(\mathcal{E}(\mathcal{C}^\dagger))$. It will be convenient to consider the following alternative encoding of CNFs $\mathcal{C}$ as a system of polynomial equations. For each variable z_i introduce two variables, denoted z_i and $\overline{z_i}$, along with the axioms

$$\forall i : z_i(1 - z_i) = 0, \quad z_i + \overline{z_i} = 1$$

which enforce that $z_i = 1 - \overline{z_i}$ and $z_i, \overline{z_i} \in \{0, 1\}$ (this encoding is typically used in the "polynomial calculus with resolution", or PCR, proof system [2]). Then encode each clause C_i as

$$\mathcal{E}^\square(C_i) = \prod_{j \in C_i^+} z_j \prod_{j \in C_i^-} \overline{z_j},$$

which yields an encoding of $\mathcal{C}$ in $\mathbf{F}[z_1, \overline{z_1}, z_2, \overline{z_2}, \ldots, z_n, \overline{z_n}]$. We show that $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}(\mathcal{C})) = \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^\square(\mathcal{C}))$ and then that $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}^\square(\mathcal{C})) = \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^\square(\mathcal{C}^\dagger))$.

First observe that $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}(\mathcal{C})) \leq \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^\square(\mathcal{C}))$ is easy: in the refutation of $\mathcal{E}^\square(\mathcal{C})$ replace every literal $\overline{z_i}$ with $1 - z_i$. So, we focus on proving $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}^\square(\mathcal{C})) \leq \mathsf{NS}_{\mathbf{F}}(\mathcal{E}(\mathcal{C}))$.

Suppose we have a Nullstellensatz refutation of $\mathcal{E}(\mathcal{C})$, and we construct a Nullstellensatz refutation of $\mathcal{E}^{\square}(\mathcal{C})$ of the same degree. For this, it suffices to show that there is a low degree proof of $\mathcal{E}(C)$ from $\mathcal{E}^{\square}(C)$ for each clause $C \in \mathcal{C}$. Write $\mathcal{E}^{\square}(C)$ as $\prod_{i\in C^+} z_i \prod_{i\in C^-} \overline{z_i}$, and we use the axioms $\overline{z_j} + z_j - 1 = 0$ for each $j \in C^+$ to derive $\mathcal{E}(C)$. To do this, multiply the axiom by $-\prod_{i\in C^-} \overline{z_i}$, yielding

$$-\prod_{i\in C^-} \overline{z_i}(\overline{z_j} + z_j - 1) = (1 - \overline{z_j}) \prod_{i\in C^-} \overline{z_i} - z_j \prod_{i\in C^-} \overline{z_i}.$$

Doing this for each $i \in C^+$ and factoring yields

$$\prod_{j\in C^+} (1 - \overline{z_j}) \prod_{j\in C^-} \overline{z_i} - \prod_{j\in C^+} z_j \prod_{j\in C^-} \overline{z_i}$$

which yields $\mathcal{E}(C)$ (over $\overline{z_i}$ variables) after adding $\mathcal{E}^{\square}(C)$. Performing this multiplication for each $C \in \mathcal{C}$ yields $\mathcal{E}(\mathcal{C})$, and it is easy to see that the degree is less than the degree of $\mathcal{E}(\mathcal{C})$.

Now let us prove $\mathcal{E}^{\square}(\mathcal{C}) = \mathcal{E}^{\square}(\mathcal{C}^{\dagger})$. Observe that if $\sum_{C\in\mathcal{C}} \mathcal{E}^{\square}(C) q_i = 1$ is a Nullstellensatz refutation of $\mathcal{E}^{\square}(\mathcal{C})$ then $\sum_{C\in\mathcal{C}} \mathcal{E}^{\square}(C^{\dagger}) q_i^{\dagger} = 1$ is a Nullstellensatz refutation of $\mathcal{C}^{\dagger}$, where $q_i^{\dagger}$ is the polynomial obtained from q_i by exchanging the variables z_i and $\overline{z_i}$ for each $i \in [n]$ and $b \in \{0, 1\}$. This shows that $\mathsf{NS}_{\mathbf{F}}(\mathcal{E}^{\square}(\mathcal{C})) = \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^{\square}(\mathcal{C}^{\dagger}))$, and thus $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbf{F}}(\mathcal{C}^{\dagger})$. □

We can now prove the upper bound for characteristic 2.

Lemma 4.3.4. *For any field* $\mathbf{F}$ *of characteristic* 2 *and any unsatisfiable CNF formula* $\mathcal{C}$, $\mathsf{gap}(\mathcal{E}(\mathcal{C})) \geq \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$.

Proof. We argue that from any d-design for $\mathcal{E}(\mathcal{C}^{\dagger})$ we can construct a polynomial p witnessing an algebraic gap of $d + 1$. By Lemma 4.3.3, $\mathcal{E}(\mathcal{C}^{\dagger})$ has an $(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) - 1)$-design, so the lemma follows.

Let D be a d-design for $\mathcal{E}(\mathcal{C}^{\dagger})$. For any $S \subseteq [n]$ we let z_S denote the monomial $\prod_{i\in S} z_i$, and thus $z_{\emptyset} = 1$. The polynomial p is defined by its coefficients: for each $S \subseteq [n]$ let $\hat{p}(S) = D(x_{[n]\setminus S})$. This immediately implies that $\deg p = n$ since $\hat{p}([n]) = D(1) = 1$ so we focus on proving that

$$\forall \pi \in \mathsf{Cert}(\mathcal{C}) : \deg(p{\restriction}\pi) \leq n - (d + 1).$$

We begin by giving an equivalent description of the previous constraint. Let $S \subseteq [n] \setminus \pi^{-1}(\{0, 1\})$ be any subset of indices, and consider the equation

$$0 = \sum_{T\subseteq\pi^{-1}(1)} \hat{p}(S \cup T). \tag{4.1}$$

We claim that if Equation 4.1 is satisfied for every certificate $\pi \in \mathsf{vars}(\mathcal{C})$ and every $S \subseteq [n] \setminus \pi^{-1}(\{0,1\})$ with $|S| \geq n-d$ then p witnesses an algebraic gap of $d+1$. To see this, let $T \subseteq [n]$ be arbitrarily chosen, and observe that after restricting the monomial z_T with π we either obtain 0, if $T \cap \pi^{-1}(0) \neq \emptyset$, or $z_{T \setminus \pi^{-1}(1)}$ otherwise. Thus, in order for $\deg(p{\restriction}\pi) \leq n-(d+1)$ all monomials of degree at least $n-d$ that remain after restricting by π must cancel; this happens if and only if Equation 4.1 holds. By definition of the coefficients of p we must therefore verify that

$$0 = \sum_{T \subseteq \pi^{-1}(1)} D(z_{[n]\setminus S \cup T})$$

Letting $U = [n] \setminus S \cup \pi^{-1}(\{0,1\})$ we can re-write this equation as

$$0 = \sum_{T \subseteq \pi^{-1}(1)} D(z_U z_{\pi^{-1}(0)} z_T).$$

Since π is a certificate of a clause $C \in \mathcal{C}$, we have $\pi^{-1}(1) = C^-$ and $\pi^{-1}(0) = C^+$ since the certificate falsifies C. By the linearity of D we can thus rewrite the previous equation as

$$\begin{aligned} 0 &= D\left(z_U z_{C^+}\left(\sum_{T \subseteq C^-} z_T\right)\right) \\ &= D(z_U \mathcal{E}(C^\dagger)), \end{aligned}$$

where the last equation follows from the definition of $\mathcal{E}(C^\dagger)$ over characteristic 2:

$$\mathcal{E}(C^\dagger) = \prod_{i \in C^+} z_i \prod_{i \in C^-}(1-z_i) = \prod_{i \in C^+} z_i \prod_{i \in C^-}(1+z_i) = z_{C^+} \sum_{T \subseteq C^-} z_T. \tag{4.2}$$

Since $|S| \geq n-d$ and $U = [n] \setminus S \cup \mathsf{vars}(C)$ we have that $|U \cup \mathsf{vars}(C)| \leq d$, and so $\deg(z_U \mathcal{E}(C^\dagger)) \leq d$, implying that $D(z_U \mathcal{E}(C^\dagger)) = 0$ by the design property, and we have shown that $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) \geq d+1$. The lemma follows. $\square$

It is natural to ask what goes wrong in the previous proof when the characteristic is not 2. In Equation 4.2 we can no longer replace $(1-z_i)$ with $(1+z_i)$, and so we obtain

$$\mathcal{E}(C^\dagger) = \prod_{i \in C^+} z_i \prod_{j \in C^-}(1-z_j) = z_{C^+}\left(\sum_{T \subseteq C^-}(-1)^{|T|} z_T\right),$$

and the alternating factor $(-1)^{|T|}$ prevents us from being able to apply the design property. In the next section we resolve this problem by a change of basis.

4.4 Algebraic Gaps = Nullstellensatz (General Case)

As discussed in the prequel, the main obstruction to proving the equivalence between algebraic gaps and Nullstellensatz degree over arbitrary fields is an "alternation" that occurs. We will resolve this problem by changing to the *Fourier basis* — which is a fancy way of saying that we move from $\{0,1\}$ assignments to $\{-1,1\}$ assignments via the affine transformation $z \mapsto 1-2z$.

Given a clause C, define the polynomial

$$\mathcal{E}^*(C) = \prod_{i \in C^+} (1+z_i) \prod_{i \in C^-} (1-z_i).$$

This is obtained from applying the affine transformation $(1-z_i)/2$ to $\mathcal{E}(C)$ and discarding the powers of 2 — observe that if we think of -1 as "True" and 1 as "False" then $\mathcal{E}^*(C) = 0$ if and only if the clause $\mathcal{C}$ is satisfied. (Clearly this transformation is only useful when the characteristic is different from 2.) Define $\mathcal{E}^*(\mathcal{C}) = \{\mathcal{E}^*(C)\}_{C \in \mathcal{C}} \cup \{z_i^2 - 1\}_{i \in [n]}$ for an unsatisfiable CNF $\mathcal{C}$. Let $\mathsf{NS}^*_{\mathbf{F}}(\mathcal{C})$ denote the Nullstellensatz degree required to refute the system $\mathcal{E}^*(\mathcal{C})$, and it is easy to see that $\mathsf{NS}^*_{\mathbf{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$ (just replace every variable z_i with $(1-z_i)/2$, or symmetrically replace z_i with $1-2z_i$).

Similarly, we define $\mathsf{gap}^*_{\mathbf{F}}(\mathcal{C})$ to be the same as $\mathsf{gap}_{\mathbf{F}}(\mathcal{C})$ except with respect to $\{-1,1\}$ restrictions. Formally speaking, for each certificate $\pi \in \mathsf{Cert}(\mathcal{C})$ one can apply the transformation $1-2z_i$ to each coordinate of π, obtaining a $\{-1,1\}$ restriction. (Observe that our initial definition of algebraic gaps from Section 4.2 was $\mathsf{gap}^*_{\mathbb{R}}(\mathcal{C})$ rather than $\mathsf{gap}_{\mathbb{R}}(\mathcal{C})$.) Once again, it is not hard to see that $\mathsf{gap}^*_{\mathbf{F}}(\mathcal{C}) = \mathsf{gap}_{\mathbf{F}}(\mathcal{C})$: given a polynomial p witnessing $\mathsf{gap}_{\mathbf{F}}(\mathcal{C})$ simply replace every variable z_i with $(1-z_i)/2$, and since the degree of polynomials is preserved under affine maps the resulting polynomial will witness $\mathsf{gap}^*_{\mathbf{F}}(\mathcal{C})$ (we can go in reverse symmetrically).

Lemma 4.4.1. *For any field* $\mathbf{F}$ *of characteristic other than* 2 *and any unsatisfiable CNF formula* $\mathcal{C}$, $\mathsf{gap}_{\mathbf{F}}(\mathcal{E}^*(\mathcal{C})) \geq \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^*(\mathcal{C}))$.

Proof. By the discussion above, we just need show that $\mathsf{gap}_{\mathbf{F}}(\mathcal{E}^*(\mathcal{C})) \geq \mathsf{NS}_{\mathbf{F}}(\mathcal{E}^*(\mathcal{C}))$. We show that if $\mathcal{E}^*(\mathcal{C}^\dagger)$ has a d-design then $\mathcal{E}^*(\mathcal{C})$ has algebraic gap complexity at least $d+1$. By Lemma 4.3.3, $\mathcal{E}^*(\mathcal{C}^\dagger)$ has an $(\mathsf{NS}_{\mathbf{F}}(\mathcal{C})-1)$-design, and so this completes the proof of the lemma.

So, let D be a d-design for $\mathcal{E}^*(\mathcal{C}^\dagger)$ and for any $S \subseteq [n]$ let z_S denote the monomial $\prod_{i \in S} z_i$. Recall from Section 4.1 that

$$\mathcal{E}^*(\mathcal{C}) = \prod_{i \in C^+} (1+z_i) \prod_{j \in C^-} (1-z_j) = \sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^-|} z_T. \tag{4.3}$$

From D we define a multilinear polynomial p witnessing algebraic gaps for $\mathcal{E}^*(\mathcal{C})$. (Note that the d-design is defined for the *dual* $\mathcal{C}^\dagger$ of $\mathcal{C}$, while the algebraic gaps are for $\mathcal{C}$.) We define the (multilinear) polynomial p by its coefficients: namely, for each $S \subseteq [n]$ let $\hat{p}(S) = D(z_{[n]\setminus S})$.

Clearly $\deg p = n$ since $\hat{p}([n]) = D(1) = 1$ so we focus on proving that $\deg(p{\restriction}\pi) \leq n - (d+1)$ for all certificates $\pi \in \mathsf{Cert}(\mathcal{E}^*(\mathcal{C}))$.

This condition is equivalent to the following system of linear equations on the coefficients of $\hat{p}$: for any clause C and any subset $S \subseteq [n]$ with $S \cap \mathsf{vars}(C) = \emptyset$ and $|S| \geq n - d$ we have

$$0 = \sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} \hat{p}(S, T). \tag{4.4}$$

By the definition of p, to finish the proof we must verify that

$$0 = \sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} D(z_{[n]\setminus(S\cup T)}).$$

Letting $U = [n] \setminus (S \cup \mathsf{vars}(C))$ we can re-write this equation as

$$0 = \sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} D(z_U z_{\mathsf{vars}(C)\setminus T}).$$

Observing that $(-1)^{|T\cap C^+|}(-1)^{|(\mathsf{vars}(C)\setminus T)\cap C^+|} = (-1)^{|C^+|}$, the linearity of D and Equation 4.3 implies that

$$\begin{aligned}
0 &= \sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} D(z_U z_{\mathsf{vars}(C)\setminus T}) \\
&= D\left(\sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} z_U z_{\mathsf{vars}(C)\setminus T} \right) \\
&= D\left(z_U \left(\sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|T \cap C^+|} z_{\mathsf{vars}(C)\setminus T} \right)\right) \\
&= D\left(z_U \left(\sum_{T \subseteq \mathsf{vars}(C)} (-1)^{|C^+|}(-1)^{|(\mathsf{vars}(C)\setminus T)\cap C^+|} z_{\mathsf{vars}(C)\setminus T} \right)\right) = (-1)^{|C^+|} D(z_U \mathcal{E}^*(C^\dagger)).
\end{aligned}$$

Since $|S| \geq n - d$ and $U = [n] \setminus (S \cup \mathsf{vars}(C^\dagger))$ we have that $|U \cup \mathsf{vars}(C^\dagger)| \leq |[n] \setminus S| \leq d$, and so $\deg(z_U \mathcal{E}^*(C^\dagger)) \leq d$, implying that $D(z_U \mathcal{E}^*(C^\dagger)) = 0$ by the design property. □

Theorem 4.3.1 follows immediately from Lemma 4.3.4 and Lemma 4.4.1.

Chapter 5

Main Lifting Theorems

We are now ready to proceed in full generality. The following theorem is the goal of this chapter, and is the central contribution of this thesis.

Theorem 5.0.1. *Let $\mathcal{C}$ be an unsatisfiable, bounded-width CNF on n variables, and let $\mathbf{F}$ be any field. Let $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be any good gadget over $\mathbf{F}$ with $|\mathcal{X}| = O(\mathrm{rank}(g))$, and consider the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$. Then*

1. *If $\mathrm{rank}(g) = n^2$ then $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}), \chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) = n^{\Theta(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))}$.*
2. *If $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) \geq \varepsilon n$ for some universal ε and $\mathrm{rank}(g) = 2^{1/\varepsilon}+1$, then $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}), \chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) = 2^{\Theta(n)}$.*

This improves our "warm-up" Theorem 3.3.2 in several ways. First, it replaces the use of Sherstov's specific gadget with any "good" gadget — in particular, this allows the lifting theorem to work for arbitrary fields instead of just the real numbers. Second, it uses the connection from the previous chapter (showing that algebraic gaps is the same as Nullstellensatz degree) to obtain bounds in terms of Nullstellensatz instead of algebraic gaps. Finally, where Theorem 3.3.2 only obtained *lower* bounds on the rank measure in terms of the algebraic gaps, Theorem 5.0.1 obtains nearly tight upper and lower bounds on both the rank measure and the monotone span program size in terms of Nullstellensatz degree.

The theorem is proved in three steps. We begin by generalizing Sherstov's rank theorem for pattern matrices to arbitrary fields. Our proof has several advantages beyond just working for arbitrary fields: first, it is much simpler, requiring only elementary notions from algebra; second, we can give a generic sufficient condition on the gadgets rather than working with a specific gadget g.

Second, using this result and the general tools we developed in the previous chapter, we prove a generalization of our "warmup" algebraic-gaps to rank-measure theorem (cf. Theorem 3.3.2) which works for arbitrary fields. By using the fact that algebraic gaps are equal to Nullstellensatz degree (cf. Theorem 4.3.1) we obtain the lower bound in Theorem 5.0.1.

For the upper bound we prove a second lifting theorem directly from the Nullstellensatz degree to the *algebraic tiling number*. By combining this with our lifting theorem for the rank measure, we will be able to give strong separations between span programs over different fields.

5.1 Lifting Polynomial Degree to Rank

In this section we generalize Sherstov's rank theorem for pattern matrices [61] (cf. Theorem 3.1.2). Before we state and prove the theorem, we first state our sufficient condition on gadgets which enables this rank "lifting". If $g : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ is a gadget then $\mathrm{rank}_{\mathbf{F}}(g)$ is the rank of g interpreted as an $\mathcal{X} \times \mathcal{Y}$ matrix over the field $\mathbf{F}$. Further, recall that if A is an $m \times n$ matrix and B is a $p \times q$ matrix then the *Kronecker product* (also called the *tensor product*) $A \otimes B$ is the $mp \times nq$ matrix defined by $(A \otimes B)((i,k),(j,\ell)) = A(i,j)B(k,\ell)$

Definition 5.1.1. Let $\mathbf{F}$ be a field. A gadget $g : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ is *good* if, for every pair of matrices A, B over $\mathbf{F}$ of the same size we have

$$\mathrm{rank}(\mathbb{1} \otimes A + g \otimes B) = \mathrm{rank}(A) + \mathrm{rank}(g)\,\mathrm{rank}(B).$$

For some intuition for good gadgets, note that rank is well-known to be multiplicative under Kronecker products: $\mathrm{rank}(A \otimes B) = \mathrm{rank}(A)\,\mathrm{rank}(B)$. However, while rank is *subadditive* under matrix addition, it is far from being *additive*: for example, taking the identity matrix I we have $\mathrm{rank}(I) = \mathrm{rank}(-I) = n$, but $\mathrm{rank}(I + (-I)) = 0$. A gadget g is therefore good if tensoring with g forces rank to behave additively.

While this property is strange at first, it turns out that Sherstov's gadget from Theorem 3.1.2 is good for all fields of characteristic other than 2. We prove this in the next section, as well as introducing a different gadget that is good for *all* fields.

Theorem 5.1.2. *Let $\mathbf{F}$ be any field and let $p \in \mathbf{F}[z_1, z_2, \ldots, z_n]$ be a multilinear polynomial over $\mathbf{F}$. For any good gadget $g : \mathcal{X} \times \mathcal{Y} \to \mathbf{F}$ we have*

$$\mathrm{rank}_{\mathbf{F}}(p \circ g^n) = \sum_{S : \hat{p}(S) \neq 0} \mathrm{rank}_{\mathbf{F}}(g)^{|S|}$$

where $\hat{p}(S)$ denotes the coefficient of the monomial $\prod_{i \in S} z_i$ in p.

Proof. The theorem follows by an easy induction using the following claim.

Claim. Let $S \subseteq [n]$, and let $z_S = \prod_{i \in S} z_i$ denote a monomial over $\mathbf{F}[z_1, z_2, \ldots, z_n]$. Then

$$z_S \circ g^n = \bigotimes_{i=1}^{n} M_S(i)$$

where $M_S(i) = g$ if $i \in S$ and $M_S(i) = \mathbb{1}$ otherwise.

Proof of Claim. For notational simplicity suppose that $S = \{1, 2, \ldots, t\}$ for some $t \leq n$, and a symmetric calculation applies for general S. Then

$$\begin{aligned} z_S \circ g^n &= [z_S(g(x_1, y_1), g(x_2, y_2), \cdots, g(x_n, y_n))]_{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n} \\ &= \left[\prod_{i \in S} g(x_i, y_i)\right]_{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n} \\ &= [g(x_1, y_1) g(x_2, y_2) \cdots g(x_t, y_t) \mathbb{1}(x_{t+1}, y_{t+1}) \cdots \mathbb{1}(x_n, y_n)]_{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n} \\ &= \underbrace{g \otimes g \otimes \cdots \otimes g}_{t \text{ times}} \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{n-t \text{ times}} \\ &= \bigotimes_{i=1}^{n} M_S(i), \end{aligned}$$

where we have used the definition of the Kronecker product. □

For the induction, we recall that the Kronecker product is bilinear: if A, B, C, D are matrices then

$$(A + B) \otimes C = A \otimes C + B \otimes C$$

and

$$A \otimes (C + D) = A \otimes C + A \otimes D$$

whenever the sums are well-defined. We now prove the theorem by induction on n.

When $n = 0$ the polynomial p is just a constant $\hat{p}(\emptyset)$ in $\mathbf{F}$, and the matrix $p \circ g^0$ is the 1×1 matrix $[\hat{p}(\emptyset)]$. In this case the conclusion of the theorem is trivially satisfied — if $\hat{p}(\emptyset) = 0$ then $\operatorname{rank}(p \circ g^0) = 0$ and if $\hat{p}(\emptyset) \neq 0$ then $\operatorname{rank}(p \circ g^0) = 1 = \operatorname{rank}(g)^0$.

Now, suppose that $n > 0$, and write $p = q + z_1 r$, where $q, r \in \mathbf{F}[z_2, z_3, \ldots, z_n]$. By the claim and the bilinearity of the Kronecker product, we write

$$\begin{aligned}
p \circ g^n &= \sum_{S:\hat{p}(S)\neq 0} \hat{p}(S) \bigotimes_{i=1}^{n} M_S(i) \\
&= \sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\notin S}} \hat{p}(S)\cdot \mathbf{1} \otimes \bigotimes_{i=2}^{n} M_S(i) + \sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\in S}} \hat{p}(S)\cdot g \otimes \bigotimes_{i=2}^{n} M_S(i) \\
&= \mathbf{1} \otimes \left(\sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\notin S}} \hat{p}(S) \bigotimes_{i=2}^{n} M_S(i) \right) + g \otimes \left(\sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\in S}} \hat{p}(S) \bigotimes_{i=2}^{n} M_S(i) \right) \\
&= \mathbf{1} \otimes (q \circ g^{n-1}) + g \otimes (r \circ g^{n-1}).
\end{aligned}$$

Applying the inductive assumption and using that g is good, we have

$$\begin{aligned}
\operatorname{rank}_{\mathbf{F}}(p \circ g^n) &= \operatorname{rank}_{\mathbf{F}}(\mathbf{1} \otimes (q \circ g^{n-1}) + g \otimes (r \circ g^{n-1})) \\
&= \operatorname{rank}_{\mathbf{F}}(q \circ g^{n-1}) + \operatorname{rank}_{\mathbf{F}}(g) \operatorname{rank}_{\mathbf{F}}(r \circ g^{n-1}) \\
&= \sum_{T:\hat{q}(T)\neq 0} \operatorname{rank}_{\mathbf{F}}(g)^{|T|} + \sum_{T:\hat{r}(T)\neq 0} \operatorname{rank}_{\mathbf{F}}(g)^{|T|+1} \\
&= \sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\notin S}} \operatorname{rank}_{\mathbf{F}}(g)^{|S|} + \sum_{\substack{S:\hat{p}(S)\neq 0\\ 1\in S}} \operatorname{rank}_{\mathbf{F}}(g)^{|S|} = \sum_{S:\hat{p}(S)\neq 0} \operatorname{rank}_{\mathbf{F}}(g)^{|S|},
\end{aligned}$$

where we note that in the above sums $T \subseteq \{2, 3, \dots, n\}$ and $S \subseteq [n]$. □

5.2 Constructing Good Gadgets

In this section we describe a simple good gadget that we can use for every field, and also observe that the gadget used by Sherstov [61] is good for all fields of characteristic other than 2. We first describe our new simple gadget that is good over every field.

Definition 5.2.1. For any positive integer λ define $h_\lambda : [\lambda+1] \times [\lambda+1] \to \{0,1\}$ by

$$h_\lambda(x,y) = \begin{cases} 1 & \text{if } x = y = i \text{ for some } i \in [\lambda] \\ 0 & \text{otherwise.} \end{cases}$$

That is, h_λ is the $(\lambda+1)\times(\lambda+1)$ identity matrix with one of the 1s deleted.

Lemma 5.2.2. *For any positive integer λ and any field $\mathbf{F}$ the gadget h_λ is good and satisfies* $\operatorname{rank}(h_\lambda) = \lambda$.

Proof. Clearly $\mathrm{rank}(h_\lambda) = \lambda$, so we focus on the additivity of rank. By definition of $\mathbb{1}$, h_λ, and the Kronecker product, we have

$$\mathbb{1} \otimes A = \begin{pmatrix} A & A & \cdots & A \\ A & A & \cdots & A \\ & & \vdots & \\ A & A & \cdots & A \end{pmatrix} \qquad h_\lambda \otimes B = \begin{pmatrix} B & 0 & \cdots & 0 & 0 \\ 0 & B & \cdots & 0 & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & B & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Adding them yields the matrix

$$\begin{pmatrix} A+B & A & \cdots & A & A \\ A & A+B & \cdots & A & A \\ & & \vdots & & \\ A & A & \cdots & A+B & A \\ A & A & \cdots & A & A \end{pmatrix},$$

which is easily verified to be row- and column-equivalent to

$$\begin{pmatrix} B & 0 & \cdots & 0 & 0 \\ 0 & B & \cdots & 0 & 0 \\ & & \vdots & & \\ 0 & 0 & \cdots & B & 0 \\ 0 & 0 & \cdots & 0 & A \end{pmatrix},$$

by subtracting the final column from each other column, and then subtracting the last row from all other rows. The rank property follows since there are λ copies of B on the diagonal. $\square$

Next, we observe that the gadget introduced by Sherstov [61] (that is, the gadget used in Theorem 3.1.2) is good for all fields of characteristic other than 2. We do not need this result as we can use the gadget h_λ for every field, and thus the reader can safely skip to the next section if they wish. However, we believe it is a nice observation that the goodness of g_λ combined with Theorem 5.1.2 yields an alternative proof of Theorem 3.1.2.

Definition 5.2.3 (Implicit in [61])**.** For any positive integer λ, define the gadget $g_\lambda : ([\lambda] \times \{-1,1\}) \times \{-1,1\}^\lambda \to \{-1,1\}^\lambda$ by $g_\lambda((x,b),y) = by_x$.

Lemma 5.2.4. *For any positive integer λ and any field $\mathbf{F}$ of characteristic other than 2, the gadget g_λ is good and satisfies $\mathrm{rank}(g_\lambda) = \lambda$.*

Proof. By definition of g_λ it should be clear that g_λ is equivalent (up to a permutation of rows) to the matrix

$$\begin{pmatrix} F_\lambda \\ -F_\lambda \end{pmatrix}$$

where $F_\lambda : [\lambda] \times \{\pm 1\}^\lambda$ is defined by $F_\lambda(x, y) = y_x$. Note that F_λ is full rank (that is, it has rank λ), implying that $\mathrm{rank}_{\mathbf{F}}(g_\lambda) = \mathrm{rank}_{\mathbf{F}}(F_\lambda) = \lambda$.

By the definition of the Kronecker product

$$\mathbb{1}_{2\lambda,2^\lambda} \otimes A + g \otimes B = \mathbb{1}_{2\lambda,2^\lambda} \otimes A + \begin{pmatrix} F_\lambda \otimes B \\ -F_\lambda \otimes B \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{\lambda,2^\lambda} \otimes A + F_\lambda \otimes B \\ \mathbb{1}_{\lambda,2^\lambda} \otimes A - F_\lambda \otimes B \end{pmatrix}.$$

By adding the top half of the last matrix to the bottom half, we obtain

$$\begin{pmatrix} \mathbb{1}_{\lambda,2^\lambda} \otimes A + F_\lambda \otimes B \\ 2 \cdot \mathbb{1}_{\lambda,2^\lambda} \otimes A \end{pmatrix},$$

from which we get the matrix

$$\begin{pmatrix} F_\lambda \otimes B \\ \mathbb{1}_{\lambda,2^\lambda} \otimes A \end{pmatrix}$$

after dividing the bottom half by 2 and then subtracting it from the top half. Since the Kronecker product is multiplicative with respect to rank we have that $\mathrm{rank}(F_\lambda \otimes B) = \lambda \cdot \mathrm{rank}(B)$, and thus there exists a sequence of row operations that can be applied to the matrix $F_\lambda \otimes B$ to obtain the matrix $(I_{\lambda \cdot \mathrm{rank}(B)} \ \mathbf{0})$ where $\mathbf{0}$ is a block matrix of 0s. Applying these row operations to the top half of the previous matrix we obtain

$$\begin{pmatrix} I_{\lambda \cdot \mathrm{rank}(B)} & \mathbf{0} \\ \mathbb{1}_{\lambda,2^{\lambda-1}} \otimes A & \mathbb{1}_{\lambda,2^{\lambda-1}} \otimes A \end{pmatrix}.$$

The rank of this matrix is clearly $\mathrm{rank}_{\mathbf{F}}(A) + \lambda \cdot \mathrm{rank}_{\mathbf{F}}(B) = \mathrm{rank}_{\mathbf{F}}(A) + \mathrm{rank}_{\mathbf{F}}(g_\lambda)\,\mathrm{rank}_{\mathbf{F}}(B)$. □

We note that one can remove some unnecessary rows and columns from the gadget g_λ to obtain a smaller gadget that is also good with the same rank.

5.3 Lifting Algebraic Gaps to Razborov's Rank Measure

We are now in a position to collect together all prior results and prove our main lifting theorem from Nullstellensatz degree to Razborov's rank measure. Its proof follows the proof of

Theorem 3.3.2 while appealing to the more general results proved in this chapter and the prequel. We are also able to optimize the lower bound when the Nullstellenstaz degree is nearly maximal — this result is required later when we prove *strongly* exponential lower bounds.

Theorem 5.3.1. *Let $\mathcal{C}$ be an unsatisfiable, polynomial-size k-CNF on n variables and $|\mathcal{C}| = n^c$ clauses, and let $\mathbf{F}$ be any field. Let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be any good gadget over $\mathbf{F}$, and consider the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$. Let $N = |\mathcal{R}_{\mathcal{C},g}| = |\mathcal{C}||\mathcal{X}|^k$ be the size of the canonical rectangle cover.*

1. *If* $\mathrm{rank}(g) = n^2$ *then*

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \geq n^{\mathsf{NS}_{\mathbf{F}}(\mathcal{C})}/6 = \frac{1}{6}\left(\frac{N}{|\mathcal{X}|^k}\right)^{\mathsf{NS}_{\mathbf{F}}(\mathcal{C})/c}.$$

2. *If* $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) \geq \varepsilon n$ *for some universal* ε *then* $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \geq 2^{(\varepsilon \log \mathrm{rank}(g) - 1)n}$.

Proof. We remark that the proof is essentially identical to the proof of our "warmup" (Theorem 3.3.2). First suppose that $\mathrm{rank}(g) = n^2$. Let $p \in \mathbf{F}[z_1, z_2, \ldots, z_n]$ be the polynomial witnessing the algebraic gap complexity $\mathsf{gap}_{\mathbf{F}}(\mathcal{C})$, and let $A = p \circ g^n$ be the pattern matrix obtained by composing p and g. We prove

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}, A) = \frac{\mathrm{rank}_{\mathbf{F}}(A)}{\max\limits_{R \in \mathcal{R}_{\mathcal{C},g}} \mathrm{rank}_{\mathbf{F}}(A{\upharpoonright}R)} \geq \Omega(n^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C})}),$$

and the result follows since $\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) = \mathsf{NS}_{\mathbf{F}}(\mathcal{C})$ by Theorem 4.3.1. Let us first analyze the denominator. Let $R_{C,\alpha}$ be an arbitrary rectangle from the cover $\mathcal{R}_{\mathcal{C},g}$ and let $\pi = \mathsf{Cert}(C)$. We claim that

$$\mathrm{rank}_{\mathbf{F}}(A{\upharpoonright}R_{C,\alpha}) = \sum_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} \mathrm{rank}(g)^{|S|}, \tag{5.1}$$

To prove this, we claim that the matrix $A{\upharpoonright}R_{C,\alpha}$ is column-equivalent to the block matrix

$$[(p{\upharpoonright}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}, (p{\upharpoonright}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}, \ldots, (p{\upharpoonright}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}]$$

for some number of copies of the matrix $(p{\upharpoonright}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}$. Equation 5.1 immediately follows from the claim as

$$\mathrm{rank}_{\mathbf{F}}(A{\upharpoonright}R_{C,\alpha}) = \mathrm{rank}_{\mathbf{F}}((p{\upharpoonright}\pi) \circ g^{[n]\setminus\mathsf{vars}(C)}) = \sum_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} \mathrm{rank}(g)^{|S|}$$

by Theorem 5.1.2. So let us prove the second claim.

By the definition, for all $(x, y) \in R_{C,\alpha}$ we have that $g^{\mathsf{vars}(C)}(x_{\mathsf{vars}(C)}, y_{\mathsf{vars}(C)}) = \pi$ and $x_{\mathsf{vars}(\pi)} = \alpha$. Let us first fix any assignment β to $y_{\mathsf{vars}(C)}$ so that $g^{\mathsf{vars}(C)}(\alpha, \beta) = \pi$. Then for all $(x, y) \in R_{C,\alpha}$ such that $y_{\mathsf{vars}(C)} = \beta$ we have

$$\begin{aligned} g^n(x, y) &= g^{\mathsf{vars}(C)}(\alpha, \beta) g^{[n] \setminus \mathsf{vars}(C)}(x_{[n] \setminus \mathsf{vars}(C)}, y_{[n] \setminus \mathsf{vars}(C)}) \\ &= \pi g^{[n] \setminus \mathsf{vars}(C)}(x_{[n] \setminus \mathsf{vars}(C)}, y_{[n] \setminus \mathsf{vars}(C)}), \end{aligned}$$

and so ranging $x_{[n] \setminus \mathsf{vars}(C)}, y_{[n] \setminus \mathsf{vars}(C)}$ over all values yields the matrix $(p{\upharpoonright}\pi) \circ g^{[n] \setminus \mathsf{vars}(C)}$. Then, ranging $y_{\mathsf{vars}(C)}$ over all β such that $g^{\mathsf{vars}(C)}(\alpha, \beta) = \pi$ yields the claim and Equation 5.1.

Now, consider the rank measure $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g})$, which by Theorem 5.1.2 and Equation 5.1 satisfies

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \geq \frac{\operatorname{rank}_{\mathbf{F}}(A)}{\max\limits_{R \in \mathcal{R}_{\mathcal{C},g}} \operatorname{rank}_{\mathbf{F}}(A{\upharpoonright}R)} = \frac{\sum\limits_{S : \hat{p}(S) \neq 0} \operatorname{rank}(g)^{|S|}}{\max\limits_{\pi \in \mathsf{Cert}(\mathcal{C})} \sum\limits_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} \operatorname{rank}(g)^{|S|}}$$

First assume that $\operatorname{rank}(g) = n^2$. By definition of $\mathsf{gap}_{\mathbf{F}}(\mathcal{C})$ we have $\deg p = n$ and thus $\sum_{S : \hat{p}(S) \neq 0} n^{2|S|} \geq n^{2n}$. For the denominator, since p witnesses the algebraic gap of $\mathcal{C}$, we have that $\deg p{\upharpoonright}\pi \leq n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C})$ for all $\pi \in \mathsf{Cert}(\mathcal{C})$. We may clearly assume that $\hat{p}(S) = 0$ when $|S| < n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C})$, so, for any π:

$$\begin{aligned} \sum_{S : \widehat{p{\upharpoonright}\pi}(S) \neq 0} n^{2|S|} &\leq \sum_{i=0}^{k} \binom{n}{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} n^{2(n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i)} \\ &\leq \sum_{i=0}^{k} \left(\frac{en}{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \right)^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} n^{2(n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i)} \\ &= \sum_{i=0}^{k} \left(\frac{e}{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \right)^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} n^{2n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \\ &\leq n^{2n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C})} \sum_{i=0}^{k} \left(\frac{e}{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \right)^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \leq 6 n^{2n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C})} \end{aligned}$$

since $e + (e/2)^2 + (e/3)^3 + \cdots \leq 6$. Putting it all together, we get

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}, A) \geq \frac{n^{2n}}{6 n^{2n - \mathsf{gap}_{\mathbf{F}}(\mathcal{C})}} = n^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C})}/6,$$

proving the theorem.

Now assume that $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) = \varepsilon n$ for a universal constant ε. By Equation 5.1,

$$\sum_{S : \widehat{p\restriction\pi}(S) \neq 0} \operatorname{rank}(g)^{|S|} \leq \sum_{i=0}^{k} \binom{n}{\mathsf{gap}_{\mathbf{F}}(\mathcal{C}) - i} \cdot \operatorname{rank}(g)^{n-\mathsf{gap}_{\mathbf{F}}(\mathcal{C})-i} \leq 2^n \cdot \operatorname{rank}(g)^{n-\mathsf{gap}_{\mathbf{F}}(\mathcal{C})}.$$

Then

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \geq \frac{\operatorname{rank}(g)^n}{2^n \cdot \operatorname{rank}(g)^{n-\mathsf{gap}_{\mathbf{F}}(\mathcal{C})}} = \frac{\operatorname{rank}(g)^{\mathsf{gap}_{\mathbf{F}}(\mathcal{C})}}{2^n} \geq 2^{(\varepsilon \log \operatorname{rank}(g) - 1)n} = 2^{\Omega(n)}. \qquad \square$$

5.4 Lifting Nullstellensatz to Algebraic Tiling

We now complement the lower bounds in Theorem 5.3.1 with nearly tight upper bounds in the case that $\mathcal{C}$ is bounded-width.

Theorem 5.4.1. *Let $\mathcal{C}$ be an unsatisfiable width-k CNF on n variables and let* $\mathbf{F}$ *be any field. Let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a good gadget, and consider the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$. Then*

1. *If* $\operatorname{rank}(g) = O(\mathsf{poly}(n))$ *then* $\chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \leq |\mathcal{C}||\mathcal{X}|^{k+1} n^{O(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))}$.

2. *If* $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) \leq \varepsilon n$ *for some universal* ε *and* $\operatorname{rank}(g) \leq 2^{1/\varepsilon} + 1$ *then* $\chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \leq |\mathcal{C}||\mathcal{X}|^{k+1} 2^{O(n)}$.

Proof. Consider a minimum-degree Nullstellensatz refutation of $\mathcal{E}(\mathcal{C})$ as witnessed by polynomials $q_1, q_2, \ldots, q_m$. By the definition of a Nullstellensatz refutation we immediately have that $\sum_{i=1}^{m} (p_i q_i) \circ g^n = \mathbb{1}$, where $\mathbb{1}$ is the $\mathcal{X}^n \times \mathcal{Y}^n$ all-1s matrix. However, this is not an algebraic tiling since the matrices $p_i q_i \circ g^n$ are not necessarily embedded in the rectangles $R_{C,\alpha}$. We avoid this problem by breaking up the matrices $p_i q_i \circ g^n$ into rectangles.

For each clause C_i in $\mathcal{C}$ let $p_i = \mathcal{E}(C_i)$ be the encoding of C_i in $\mathcal{E}(\mathcal{C})$. Observe that for each p_i and each $z \in \{0,1\}^n$ we have $p_i(z) \neq 0$ if and only if z is consistent with the certificate of C_i; by extension, for all $(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n$ we have that $p_i q_i \circ g^n(x,y) = 0$ unless $g^n(x,y)$ is consistent with the certificate of p_i. Thus we can write

$$\mathbf{1} = \sum_{i=1}^{m} p_i q_i \circ g^n = \sum_{i=1}^{m} \sum_{\alpha \in \mathcal{X}^{\mathsf{vars}(C_i)}} (p_i q_i \circ g^n)\restriction R_{C_i,\alpha}.$$

Since $\operatorname{rank}(g) = \mathsf{poly}(n)$, by Theorem 5.1.2

$$\operatorname{rank}_{\mathbf{F}}((p_i q_i \circ g^n)\restriction R_{C_i,\alpha}) \leq \operatorname{rank}_{\mathbf{F}}(p_i q_i \circ g^n) = \sum_{S : \widehat{p_i q_i}(S) \neq 0} \operatorname{rank}(g)^{|S|} \leq n^{O(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))}$$

for all $i \in [m]$ and $\alpha \in \mathcal{X}^{\mathsf{vars}(C_i)}$. This immediately yields

$$\chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) \leq |\mathcal{C}||\mathcal{X}^{\leq k}|n^{O(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))}.$$

An analogous calculation holds if $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) = \Theta(n)$. $\square$

With this, Theorem 5.0.1 is an easy corollary.

Proof of Theorem 5.0.1. Let $\mathcal{C}$ be a width-k unsatisfiable CNF on n variables and let $\mathbf{F}$ be any field. In both cases, the lower bound holds by applying Theorem 5.3.1. The upper bound follows from Theorem 5.4.1 since $|\mathcal{C}| \leq n^{O(k)}$ and $|\mathcal{X}| = O(\mathrm{rank}(g))$. To see this, observe that in the first case $k = O(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))$ and in the second case $k = O(1)$ implies $n^{O(k)}$ is $O(\mathsf{poly}(n))$. $\square$

Chapter 6

Applications

In this chapter we describe the applications of our main theorem, recalled here.

Theorem 5.0.1. *Let $\mathcal{C}$ be an unsatisfiable, bounded-width CNF on n variables, and let $\mathbf{F}$ be any field. Let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be any good gadget over $\mathbf{F}$ with $|\mathcal{X}| = O(\mathrm{rank}(g))$, and consider the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$. Then*

1. *If $\mathrm{rank}(g) = n^2$ then $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}), \chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) = n^{\Theta(\mathsf{NS}_{\mathbf{F}}(\mathcal{C}))}$.*
2. *If $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) \geq \varepsilon n$ for some universal ε and $\mathrm{rank}(g) = 2^{1/\varepsilon}+1$, then $\mu_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}), \chi_{\mathbf{F}}(\mathcal{R}_{\mathcal{C},g}) = 2^{\Theta(n)}$.*

By applying Proposition 3.2.3 we can convert the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$ into partial monotone boolean functions for the purpose of applying Theorem 5.0.1. In general, this monotone boolean function has a natural interpretation as a monotone variant of the SAT problem. This has essentially been discovered and re-discovered by many authors — Raz and McKenzie observed the connection for a particular $\mathcal{C}$ and g [49], and the modern interpretation was essentially found by both Göös and Pitassi [28] and Oliveira [45]. We begin this chapter by recalling the interpretation of $\mathcal{R}_{\mathcal{C},g}$ as a SAT problem. Using this interpretation and Theorem 5.0.1, we will then prove lower bounds on the rank measure for several natural computational problems.

6.1 Canonical Rectangle Covers and Monotone CSP-SAT

We begin by introducing a very general form of the constraint satisfaction problem.

Definition 6.1.1. A *constraint satisfaction problem* (CSP) is defined by a triple $\mathcal{J} = (H, \Sigma, \mathcal{P})$ where $H = (L \cup R, E)$ is a bipartite graph, Σ is a finite alphabet, and

$$\mathcal{P} = \left\{ P_r : \Sigma^{N(r)} \to \{0,1\} \mid r \in R \right\}$$

defines for each vertex $r \in R$ a predicate P_r on the neighbourhood $N(r) \subseteq L$ of r. The instance $\mathcal{J}$ is *satisfiable* if there is a Σ-valued assignment $L \to \Sigma$ to the vertices in L such that each predicate P_r is satisfied when evaluated on r's neighbourhood in L.

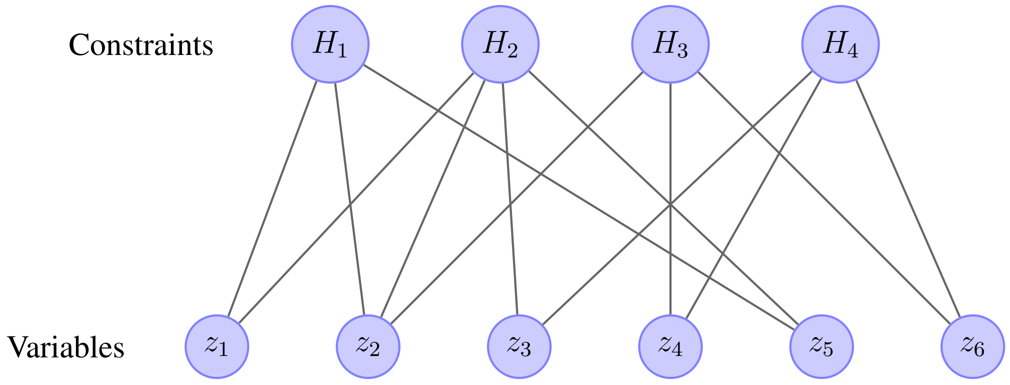

Figure 6.1: A Constraint Satisfaction Problem. The variables $z_1, z_2, \ldots, z_6$ are Σ-valued, and each H_i is an arbitrary predicate defined on its neighbourhood. The CSP is *satisfiable* if there is a Σ-valued assignment to the variables so that each predicate is satisfied.

Using constraint satisfaction problems we define the monotone variant of SAT. It is obtained by fixing H and Σ in the definition of a CSP, and letting the input string w define the constraints $\mathcal{P}$. We then accept w if and only if the CSP encoded by w is satisfiable.

Definition 6.1.2. Let $H = (L \cup R, E)$ be a bipartite graph, let Σ be a finite alphabet, and let $N = \sum_{r \in R} |\Sigma|^{|N(r)|}$. The *CSP-SAT* problem $\mathsf{SAT}_{H,\Sigma} : \{0,1\}^N \to \{0,1\}$ is defined as follows. Each $w \in \{0,1\}^N$ defines a set of predicates $\mathcal{P}_w = \left\{P_r : \Sigma^{N(r)} \to \{0,1\} \mid r \in R\right\}$ by recording a value $P_r(\alpha) \in \{0,1\}$ for each $r \in R$ and each $\alpha \in \Sigma^{N(r)}$. Then, given w, $\mathsf{SAT}_{H,\Sigma}(w) = 1$ if and only if the CSP-SAT instance $(H, \Sigma, \mathcal{P}_w)$ is satisfiable.

For any Σ, H observe that $\mathsf{SAT}_{\Sigma,H}$ is monotone since replacing a 0 with a 1 in the truth table of any constraint preserves the constraint's satisfying assignments. It should be clear from the definition of $\mathsf{SAT}_{\Sigma,H}$ that it is computable in NP.

In what remains, we will show that the partial monotone boolean function corresponding to the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$ for an unsatisfiable CNF formula $\mathcal{C}$ and gadget g can be extended to an instance of the CSP-SAT function. This is not hard, but it is technical. Let $\mathcal{C}$ be an unsatisfiable CNF with clauses $C_1, C_2, \ldots, C_m$ and variables $z_1, z_2, \ldots, z_n$, and let $\mathsf{Gr}(\mathcal{C}) := ([n] \cup [m], E)$ be the bipartite graph representing the topology of $\mathcal{C}$ (i.e. $ij \in E$ if and only if z_i appears in the clause C_j). Let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a gadget. Let us recall the definition of the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$ of $\mathcal{X}^n \times \mathcal{Y}^n$ (cf. Definition 3.2.2). For each clause C in $\mathcal{C}$ and each assignment $\alpha \in \mathcal{X}^{|C|}$ define the rectangle

$$\begin{aligned} R_{C,\alpha} &= \big\{(x,y) \in \mathcal{X}^n \times \mathcal{Y}^n \mid x_{\mathsf{vars}(C)} = \alpha \text{ and } g^n(x,y) \text{ falsifies } C\big\} \\ &= \big\{x \in \mathcal{X}^n \mid x_{\mathsf{vars}(C)} = \alpha\big\} \times \big\{y \in \mathcal{Y}^n \mid g^{\mathsf{vars}(C)}(\alpha, y_{\mathsf{vars}(C)}) \text{ falsifies } C\big\}. \end{aligned}$$

Then $\mathcal{R}_{\mathcal{C},g}$ contains all rectangles $R_{C,\alpha}$ that are not empty; it is a rectangle cover of $\mathcal{X}^n \times \mathcal{Y}^n$ since $\mathcal{C}$ is unsatisfiable. Our plan is to show that the CSP-SAT function corresponding to $\mathcal{R}_{\mathcal{C},g}$ is $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$. We do this by mapping each $x \in \mathcal{X}^n$ to an accepting instance of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$ (in fact, a *minterm* of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$) and each $y \in \mathcal{Y}^n$ to a rejecting instance of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$.

Accepting Inputs $\mathcal{U}$. Map each $x \in \mathcal{X}^n$ into the CSP with topology $\mathsf{Gr}(\mathcal{C})$ having x as its unique satisfying assignment. Formally, for each constraint vertex $j \in [m]$ and for each $\alpha \in \mathcal{X}^{N(j)}$, $P_j(\alpha) = 0$ unless $\alpha = x_{\mathsf{vars}(C_j)}$. Clearly this yields *every* minterm of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$.

Rejecting Inputs $\mathcal{V}$. Map each $y \in \mathcal{Y}^n$ into the CSP $\mathcal{H}(x) := \mathcal{C}(g^n(x,y))$. That is, for fixed y, we consider the CSP which, on input $x \in \mathcal{X}^n$ is satisfied if $\mathcal{C}(g^n(x,y)) = 1$. Formally, for each constraint vertex $j \in [m]$ define $P_j : \mathcal{X}^{N(j)} \to \{0,1\}$ by

$$P_j(\alpha) = C_j(g^{\mathsf{vars}(C_j)}(\alpha, y_{\mathsf{vars}(C_j)})).$$

That is, $P_j(\alpha)$ is satisfied iff the clause C_j, evaluated on $g^{\mathsf{vars}(C_j)}(\alpha, y)$, is satisfied. Observe that this is a rejecting input since $\mathcal{C}$ is unsatisfiable.

Note that the input variables of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$ correspond to pairs (C,α) where C is a clause of $\mathcal{C}$ and $\alpha \in \mathcal{X}^{\mathsf{vars}(C)}$ is a local assignment to the x-variables of C.

Proposition 6.1.3. *Let $\mathcal{C}$ be an unsatisfiable CNF and let $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a gadget. The monotone Karchmer-Wigderson game of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$ restricted to $\mathcal{U} \times \mathcal{V}$ is exactly $\mathcal{R}_{\mathcal{C},g}$ up to the relabelling defined above.*

Proof. Let $X_{C,\alpha} = A \times B$ be the coordinate rectangle in $\mathsf{KW}^+(\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C},\mathcal{X})})$ (restricted to $\mathcal{U} \times \mathcal{V}$) that corresponds to the pair (C,α). Similarly, let $R_{C,\alpha} = S \times T$ be the corresponding rectangle in the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$. Let $z \in A$ be defined by $z = \mathcal{U}(x)$, as defined above; then $z \in A$ if and only if $x_{\mathsf{vars}(C)} = \alpha$ (and thus $x \in S$). Similarly, if $z' \in B$ is defined by $z' = \mathcal{V}(y)$, then $z' \in B$ if and only if $g^{\mathsf{vars}(C)}(\alpha, y_{\mathsf{vars}(C)})$ falsifies C (and thus $y \in T$). $\square$

To summarize this section, we give a dictionary translating the data of the canonical rectangle cover $\mathcal{R}_{\mathcal{C},g}$ into the data of $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$.

Canonical Rectangle Cover $\mathcal{R}_{\mathcal{C},g}$	CSP-SAT Function $\mathsf{SAT}_{\mathsf{Gr}(\mathcal{C}),\mathcal{X}}$
Clauses C_i in $\mathcal{C}$	Constraint vertices $i \in [m]$ in $\mathsf{Gr}(\mathcal{C})$
Variables z_i of $\mathcal{C}$	Variable vertices $j \in [n]$ in $\mathsf{Gr}(\mathcal{C})$
$\mathsf{vars}(C_i)$	Neighbourhood $N(i) \subseteq [n]$ in $\mathsf{Gr}(\mathcal{C})$
Rectangles $R_{C,\alpha}$	Input variables $u_{C,\alpha}$
$x \in \mathcal{X}^n$	CSP with unique satisfying assignment x
$y \in \mathcal{Y}^n$	Unsatisfiable CSP $\mathcal{H}(x) := \mathcal{C}(g^n(x,y))$

6.2 Induction and The ST-Connectivity Problem

We first consider the *layered st-connectivity* function. Let m, n be positive integers, and let $G_{n,m}$ denote the following directed graph with $mn+2$ vertices $V = \{v_{i,j} \mid i \in [n], j \in [m]\} \cup \{s,t\}$. We think of the vertices as being arranged in $m+2$ layers indexed by $i = 0, 1, \ldots, m+1$: layer 0 contains the vertex s, layer $m+1$ contains the vertex t, and the jth layer for $j = 1, 2, \ldots, m$ contains vertices $\{v_{i,j} \mid i \in [n]\}$. Finally, for each pair of adjacent layers L_i, L_{i+1} add all edges oriented from L_i to L_{i+1}, and note that the final graph contains $mn^2 + 2n$ edges.

With this graph in mind, the *layered st-connectivity* function $\mathrm{STCONN}_{n,m}$ is defined as follows: the input is a boolean string of length $mn^2 + 2n$ describing a subgraph of the graph $G_{n,m}$ defined above, and the function outputs 1 if and only if there is a directed path from s to t.

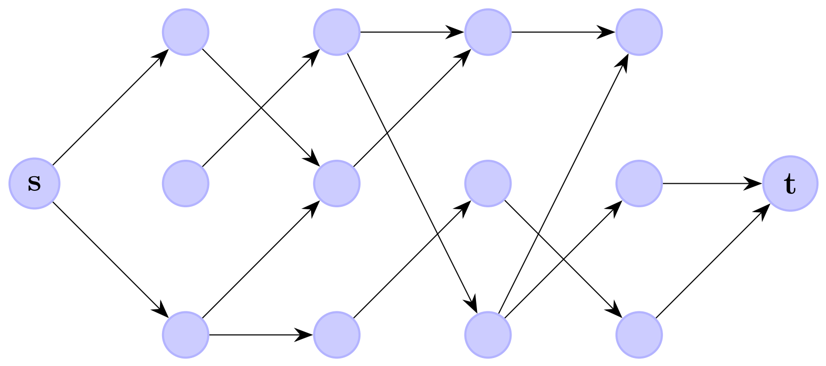

Figure 6.2: $\mathrm{STCONN}_{3,4}$: Given a subgraph of $G_{3,4}$, decide if there is a path from s to t.

In a seminal work, Karchmer and Wigderson [37] showed that optimal monotone formulas computing $\mathrm{STCONN}_{n,m}$ have size $n^{\Theta(\log m)}$ — the upper bound follows from recursive doubling, and the lower bound was shown via communication complexity; their lower bound was improved by Potechin [48] to hold for monotone switching networks. We extend this bound to the rank measure over all fields. To do this we use the following unsatisfiable CNF encoding induction, introduced by Buss and Pitassi [17], who also proved tight $\Theta(\log m)$ degree bounds on its Nullstellensatz degree.

Definition 6.2.1. Let m be a positive integer, and define Ind_m to be the unsatisfiable CNF

$$\mathsf{Ind}_m = z_1 \wedge (\overline{z}_1 \vee z_2) \wedge (\overline{z}_2 \vee z_3) \wedge \cdots \wedge (\overline{z}_{m-1} \vee z_m) \wedge \overline{z}_m.$$

Theorem 6.2.2 ([17]). *For any field* $\mathbf{F}$, $\mathsf{NS}_{\mathbf{F}}(\mathsf{Ind}_m) = \Theta(\log m)$.

We show that $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Ind}_m),\Sigma}$ is exactly the $\mathrm{STCONN}_{n,m}$ function for any alphabet Σ with $|\Sigma| = n$. Applying Theorem 5.0.1 and Theorem 6.2.2 implies that the same $n^{\Theta(\log m)}$ bounds for $\mathrm{STCONN}_{n,m}$ also hold for the rank measure over every field.

Theorem 6.2.3. *For all sufficiently large* n *and for every field* $\mathbf{F}$, $\mu_{\mathbf{F}}(\mathrm{STCONN}_{n,m}) = n^{\Theta(\log m)}$.

Proof. Let $\lambda = n^2$ and let h_λ be the good gadget from Definition 5.2.1. The upper bound follows from Theorem 2.4.4 (that is, since monotone span programs over any field can simulate monotone formulas, and $\mathsf{mSPAN}_{\mathbf{F}}(f) = \chi_{\mathbf{F}}(f) \geq \mu_{\mathbf{F}}(f)$). For the lower bound, we claim that $\mathrm{STCONN}_{\lambda+1,m}$ is a total extension of the monotone function corresponding to $\mathcal{R}_{\mathsf{Ind}_m,h_\lambda}$. Using the claim and applying Theorem 5.0.1 using the lower bound from Theorem 6.2.2 completes the theorem.

We now prove the claim. In fact, a stronger claim holds: if $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is *any* gadget with $|\mathcal{X}| = n$ then $\mathrm{STCONN}_{n,m}$ is a total extension of the partial monotone function corresponding to $\mathcal{R}_{\mathsf{Ind}_m,g}$. This stronger claim will be proved using the monotone CSP-SAT function introduced above.

Consider the accepting and rejecting inputs of $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Ind}_m),\mathcal{X}}$ obtained from $\mathcal{R}_{\mathsf{Ind}_m,g}$. We first claim that the variables of $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Ind}_m),\mathcal{X}}$ are in 1-1 correspondence with the variables of $\mathrm{STCONN}_{n,m}$. To see this, recall that each variable of $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Ind}_m),\mathcal{X}}$ corresponds to a non-empty rectangle $R_{C,\alpha} \in \mathcal{R}_{\mathcal{C},g}$, which are of the form

$$R_{C,\alpha} = \left\{x \in \mathcal{X}^n \mid x_{\mathsf{vars}(C)} = \alpha\right\} \times \left\{y \in \mathcal{Y}^n \mid g^{\mathsf{vars}(C)}(\alpha, y_{\mathsf{vars}(C)}) \text{ falsifies } C\right\}.$$

Fix such a pair C, α so that $R_{C,\alpha}$ is non-empty and suppose $C = (\overline{z_i} \vee z_{i+1})$; the input variable $u_{C,\alpha}$ corresponding to the rectangle $R_{C,\alpha}$ is therefore naturally identified with the pair $\alpha = (\alpha_1, \alpha_2) \in \mathcal{X}^2$. In $\mathrm{STCONN}_{n,m}$, the variable $u_{C,\alpha}$ thus corresponds to the edge connecting node α_1 in layer i to node α_2 in layer $i+1$. (The clauses z_1 and $\overline{z}_m$ correspond to the edges between s and the first layer and the last layer and t, respectively.)

We now claim the set $\mathcal{U}$ of accepting inputs corresponds directly to the minterms (i.e. possible st-paths) in $\mathrm{STCONN}_{n,m}$. Since this is the set of all minterms of $\mathrm{STCONN}_{n,m}$ the claim follows from Proposition 2.1.1. Fix any $x \in \mathcal{X}^m$. By the definition of $\mathcal{U}$ we will set $u_{C,\alpha} = 1$ if $x_{\mathsf{vars}(C)} = \alpha$. By our identification of edges with $x_{\mathsf{vars}(C),\alpha}$ above this corresponds to picking out one node in each layer of $\mathrm{STCONN}_{n,m}$ (namely, we choose node x_i in layer i) and adding edges connecting them all. This clearly yields a st-path, and moreover all st-paths are definable in this way since x ranges arbitrarily over $\mathcal{X}^m$. □

This gives an alternate proof of the results of Karchmer and Wigderson [37] and Potechin [48] cited above. It also provides tight superpolynomial lower bounds for both monotone comparator circuits and monotone span programs over any field computing $\mathrm{STCONN}_{n,m}$. This is notable for several reasons. Since $\mathrm{STCONN}_{n,m}$ is easily computable by polynomial-size monotone circuits, this provides the first separating example between monotone circuits and monotone span programs/monotone comparator circuits (we strengthen this separation to exponential in the next section). In fact, this shows that both of these models can be weaker than $O(\log^2 n)$-depth monotone circuits, since such circuits can compute $\mathrm{STCONN}_{n,n}$. Furthermore, Wigderson [65] proved that $\mathrm{STCONN}_{n,m}$ is computable by polynomial-size non-monotone span programs over $GF(2)$, and thus this provides the first superpolynomial separation between monotone span programs and non-monotone span programs. Similarly, Cook, Filmus and Lê [22] proved that polynomial-size non-monotone comparator circuits can compute $\mathrm{STCONN}_{n,m}$, and thus this provides the first superpolynomial separation between non-monotone comparator circuits and monotone comparator circuits.

6.3 Pebbling Tautologies and The Generation Problem

Since polynomial-size monotone circuits can compute $\mathrm{STCONN}_{n,m}$, the previous theorem yields a quasipolynomial separation between $\mu_{\mathbf{F}}$ and mP over all fields $\mathbf{F}$. We can improve this to a (weakly) exponential separation by considering the GEN function, defined next.

Let n be a positive integer and let $\mathcal{T} \subseteq [n]^3$ be a subset of triples of $[n]$. We say that $\mathcal{T}$ *generates* a point $w \in [n]$ if $w = 1$, or if there is a triple $(u, v, w) \in \mathcal{T}$ such that $\mathcal{T}$ generates u and v. The GEN_n problem is then defined as follows: as input, we receive a subset $\mathcal{T} \subseteq [n]^3$, encoded as a bitstring of length n^3, and must decide whether or not $\mathcal{T}$ generates the point n.

It is not hard to see that the GEN_n problem has polynomial-size monotone circuits, and it has been used in several previous works studying the strength of circuit classes inside mP. For instance, Raz and McKenzie [49] have used the function to separate mNC_i from mNC_{i+1} for all i, and Chan and Potechin [18] used it to prove strong lower bounds against monotone switching networks. To prove rank measure lower bounds on GEN we will use the *pebbling tautologies*, defined next.

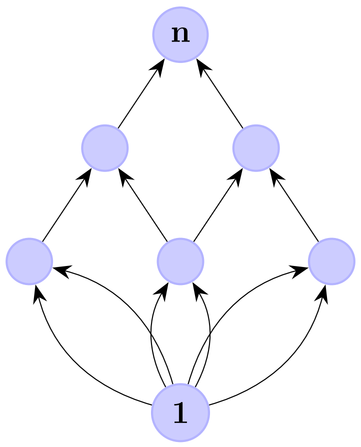

Figure 6.3: GEN_n: Given a set of triples in $[n]$, decide if we can generate n from 1.

Definition 6.3.1. Let G be a connected, directed acyclic graph with a unique sink node t and source nodes $S \subseteq V(G)$. The unsatisfiable CNF formula Peb_G is defined as follows. There is one boolean variable z_v for each vertex v in G, and we have the following clauses:

1. The *target clause* $\neg z_t$, where t is the sink node
2. For each source vertex $s \in S$ add the *source clause* z_s.
3. For each internal vertex v with in-neighbours $U \subseteq V(G)$ add the *edge clause* $\left(z_v \vee \bigvee_{u \in U} \neg z_u\right)$.

We consider pebbling tautologies where the underlying graph G is a *pyramid graph*. Let h be a positive integer. A *pyramid graph* of height h is the graph Δ_h on $n = h(h+1)/2$ vertices V, which are partitioned into h sets $V_1, V_2, \ldots, V_h$ where V_i has i vertices. Ordering each V_i as $v_{i,1}, v_{i,2}, \ldots, v_{i,i}$; then for each $i = 2, 3, \ldots, h$, if $v_{i,j}$ and $v_{i,j+1}$ are adjacent vertices in V_i add edges $(v_{i,j}, v_{i-1,j})$ and $(v_{i,j+1}, v_{i-1,j})$. A *pyramid instance* of GEN (cf. Figure 6.3) is a collection of triples $\mathcal{T}$ which is naturally isomorphic to a pyramid graph: the top point of the pyramid is n, and we assume that the point 1 is connected to each of the points $v_{1,i}$ in the first layer of the pyramid by triples $(1, 1, i)$.

Buresh-Oppenheim et al. [13] considered Nullstellensatz refutations, and showed that Nullstellensatz degree of Peb_G is tightly bounded by a combinatorial measure of directed graphs known as the *pebbling number*. Cook [21] showed that the pebbling number of the height-h pyramid Δ_h is $\Theta(h)$; combining these yields the following theorem.

Theorem 6.3.2. *[Corollary of [13] and [21]] For any positive integer h and any field* $\mathbf{F}$, $\mathsf{NS}_{\mathbf{F}}(\mathsf{Peb}_{\Delta_h}) = \Theta(h)$.

Theorem 6.3.3. *Let n be a sufficiently large positive integer. For every field* $\mathbf{F}$, $\mu_{\mathbf{F}}(\text{GEN}_n) \geq N^{\Theta(N^{\varepsilon})}$ *for some constant $\varepsilon > 0$ and N is the number of input variables to the function.*

Proof. Let k be a positive integer and let $\Delta = \Delta_k$ be the height-k pyramid graph. The tautology Peb_Δ has $m = k(k+1)$ variables; so, with an eye to applying Theorem 5.0.1, set $\lambda = m^2$ and let h_λ be the good gadget from Definition 5.2.1. By Theorem 5.0.1 and Theorem 6.3.2 we have $\mu_{\mathbf{F}}(\mathcal{R}_{\mathsf{Peb}_\Delta, h_\lambda}) = m^{\Theta(\mathsf{NS}_{\mathbf{F}}(\mathsf{Peb}_\Delta))} = m^{\Theta(k)}$. We give a monotone projection from $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Peb}_\Delta),[\lambda+1]}$ to GEN_n for $n(\lambda+1)m+2 = \Theta(k^6)$; this yields the theorem since the number of input variables to $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Peb}_\Delta),[\lambda+1]}$ is $N = \Theta(n^3)$.

The monotone projection is defined as follows. Partition the $(\lambda+1)m$ points of GEN_n into m groups $S_1, S_2, \ldots, S_m$, each of size $\lambda+1$, and further organize these m groups into a pyramid. For each $i \in [m]$, $j \in [\lambda+1]$ let $p_{i,j}$ denote the jth point in the group S_i. Label the remaining two points s and t, and they are to be thought of as the "source" and "target" point.

For each source clause $C = z_i$ and each assignment $\alpha : \{z_i\} \to [\lambda+1]$, map the variable $u_{C,\alpha}$ to the input triple $(s, s, p_{i,\alpha(i)})$. For the target clause $C = z_t$ and each assignment $\alpha : \{z_t\} \to [\lambda+1]$, map the variable $u_{C,\alpha}$ to the input triple $(p_{t,\alpha(t)}, p_{t,\alpha(t)}, t)$. Finally, for each "internal" clause $C = \neg z_i \vee \neg z_j \vee z_k$ of Peb_Δ and each assignment $\alpha : \{z_i, z_j, z_k\} \to [\lambda+1]$, map the variable $u_{C,\alpha}$ of $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Peb}_\Delta),[\lambda+1]}$ to the input triple $(p_{i,\alpha(i)}, p_{j,\alpha(j)}, p_{k,\alpha(k)})$. Fix all other inputs of GEN_n to 0 in the monotone projection.

Let $\mathcal{J}$ be a CSP corresponding to an input of $\mathsf{SAT}_{\mathsf{Gr}(\mathsf{Peb}_\Delta),[\lambda+1]}$ and let $\mathcal{T}(\mathcal{J})$ be the corresponding input to GEN_n under the above projection. We claim that $\mathcal{J}$ is satisfiable if and only if t can be generated from s in the GEN instance $\mathcal{T}(\mathcal{J})$. To see this, consider any satisfying assignment $z : [m] \to [\lambda+1]$ to the CSP $\mathcal{J}$. In the GEN instance, this corresponds to picking out one vertex $p_{i,z(i)}$ in each group of vertices S_i. Since z satisfies $\mathcal{J}$, it follows that for each constraint C the variable $u_{C,\alpha} = 1$ where $\alpha(i) = z(i)$ for each $i \in \mathsf{vars}(C)$. By the definition of the monotone projection, it follows that for each triple of vertices $p_{i,z(i)}, p_{j,z(j)}, p_{k,z(k)}$ picked out by the satisfying assignment, the corresponding triple occurs in the GEN instance (and the same holds for the "source" and "target" triples). It follows that we can generate t from s as $\mathcal{T}(\mathcal{J})$ contains a pyramid instance. The converse is shown symmetrically. $\square$

Since GEN_n is computable by polynomial-size monotone circuits, this implies an exponential separation between the rank measure over every field and monotone circuits. This proves that monotone span programs over *every* field can be weaker than monotone circuits, resolving a long-standing open problem [38]. Since there exists a monotone function computable by polynomial-size $GF(2)$-monotone span programs that requires superpolynomial-size monotone circuits [5], this shows that monotone span programs and monotone circuits are orthogonal in power. Furthermore, this yields the first exponential-size separation between monotone comparator circuits and monotone circuits, and also gives an alternative proof of the exponential-size lower bounds for monotone switching networks computing GEN_n given by Chan and Potechin [18]. Finally, we remark that by instead considering height $\log^i n$ pyramids vs. height $\log^{i-1} n$ pyramids in the proof of the above theorem we can recover the monotone depth hierarchy theorem shown by Raz and Mckenzie [49].

6.4 Counting Principles and Strongly Exponential Lower Bounds

In this section we separate the strength of monotone span programs over different fields, and also prove *strongly* exponential lower bounds (i.e. of the form $2^{\Omega(n)}$ for a function on n input variables) on $\mu_{\mathbf{F}}$.

Beimel and Weinreb [10] showed that for each prime p there is a function with polynomial size monotone span programs over $GF(p)$, but all fields with characteristic different from p require monotone span programs of size $n^{\Omega(\sqrt{\log n})}$. We improve this separation to exponential: we show that for each prime p there is a function f with polynomial-size monotone span programs over fields of characteristic p, but for all fields of characteristic $q \neq p$ the function f requires monotone span programs of exponential size. Our results are proved using the following *counting principles*.

Definition 6.4.1. Let p be a prime, and let G be a d-regular undirected graph with $n \equiv 1 \bmod p$ vertices. Define the modular counting principle $\mathsf{MOD}_{G,p}$ as follows. For each edge $uv \in E(G)$ add two "directed edge" variables z_{uv} and z_{vu} taking values in $\{0, 1, \ldots, p-1\}$. We then add the following constraints:

Edge Constraints. For each edge uv, $z_{uv} + z_{vu} \equiv 0 \bmod p$.

Vertex Constraints. For each vertex v add the constraint $\sum_{u \sim v} z_{vu} \equiv 1 \bmod p$.

We encode $\mathsf{MOD}_{G,p}$ as a k-CNF, where $k = \max\{p, d\}$, by introducing a boolean variable $\tilde{z}_{e,i}$ for each directed edge $e = uv$ and each residue $i \in \{0, 1, \ldots, p-1\}$ indicating if the edge variable z_e takes value i. Each edge and vertex constraint can then be written as a collection of $O(p^k)$ clauses on the indicator variables.

Beame et al. [6] gave degree $O(k)$ (i.e. bounded-degree, if the degree of the graph is bounded) upper-bounds for characteristic-p Nullstellensatz refutations of $\mathsf{MOD}_{G,p}$. On the other hand, Buss, Grigoriev, Impagliazzo and Pitassi [15] proved that if the graph G is a good expander then $\mathsf{MOD}_{G,p}$ is hard for Nullstellensatz refutations over any field with characteristic $q \neq p$.

Definition 6.4.2. Let G be an undirected graph. The *expansion* of G is the maximum $\varepsilon > 0$ such that for any subset S of vertices with $|S| \leq |V(G)|/2$ we have

$$|\{v \in V(G) \mid \exists u \in S \text{ such that } uv \in E\}| \geq (1+\varepsilon)|S|.$$

Theorem 6.4.3 (Theorem 15 in [15]). *Let p, q be primes, let d be a positive integer, and let $\mathbf{F}_p$ be a field with characteristic p. Let G be a d-regular undirected graph with n vertices and expansion ε. Then $\mathsf{NS}_{\mathbf{F}_p}(\mathsf{MOD}_{G,q}) \geq \varepsilon n/8$.*

Using this theorem and the fact that good (i.e. sparse and constant-degree) expanders (see, e.g. [42]) exist we can prove very strong separations over different fields.

Theorem 6.4.4. *For every prime p there exists a monotone boolean function f with N inputs such that f satisfies $\mathsf{mSPAN}_{\mathbf{F}}(f) = \mathsf{poly}(N)$ for all fields $\mathbf{F}$ of characteristic p, but for every field $\mathbf{F}'$ of characteristic $q \neq p$, $\mu_{\mathbf{F}'}(f) = 2^{\Theta(N^{\varepsilon})}$ for some universal constant ε. Furthermore, the function is f is computable in* NP.

Proof. Let $\mathbf{F}, \mathbf{F}'$ be any fields with characteristic p, q, respectively. Marcus, Spielman, and Srivistava [42] proved that for all d, n there exist d-regular graphs with n vertices and expansion $\varepsilon \geq 1/2 - 1/\sqrt{d-1}$. Fix $d = 10$ and let G be such a graph with $n \equiv 1 \bmod p$ vertices. Since G is d-regular it follows that G contains $dn/2 = 5n$ edges; thus the formula $\mathsf{MOD}_{G,p}$ contains $m = 2pn = 10pn$ variables. By the theorem just stated we have $\mathsf{NS}_{\mathbf{F}'}(\mathsf{MOD}_{G,p}) \geq \varepsilon n/8 = \varepsilon' m$ for some ε', and by the upper bound result we know that $\mathsf{NS}_{\mathbf{F}}(\mathsf{MOD}_{G,p}) = O(k)$ where $k = \max\{d, p\}$.

First set $\lambda = m^2$ and let h_λ be the good gadget from Definition 5.2.1. Using the first part of Theorem 5.0.1 we have that

$$\mu_{\mathbf{F}}(\mathcal{R}_{\mathsf{MOD}_{G,p},h_\lambda}), \chi_{\mathbf{F}}(\mathsf{MOD}_{G,p}, h_\lambda) = m^{O(k)}.$$

When considering the field $\mathbf{F}'$ with characteristic $q \neq p$, applying the first part of Theorem 5.0.1 again yields

$$\mu_{\mathbf{F}'}(\mathcal{R}_{\mathsf{MOD}_{G,p},h_\lambda}), \chi_{\mathbf{F}'}(\mathsf{MOD}_{G,p}, h_\lambda) = m^{\Theta(m)}.$$

Converting $\mathcal{R}_{\mathsf{MOD}_{G,p},h_\lambda}$ into CSP-SAT as described above proves the theorem and noting $m = \mathsf{poly}(N)$ yields the theorem, where N is the number of input variables to the resulting CSP-SAT function. □

We finish this section by applying the second part of Theorem 5.0.1 to prove *strongly* exponential lower bounds.

Theorem 6.4.5. *For every prime p and there exists a monotone boolean function f with N inputs such that for every field of characteristic different from p, $\mu_{\mathbf{F}}(f) = 2^{\Theta(N)}$. Furthermore, the function is f is computable in* NP.

Proof. Follows the proof of the previous theorem, except set $\lambda = 2^{1/\varepsilon} + 1$ where ε is the expansion and use the second part of Theorem 5.0.1. □

Thus for every model lower-bounded by the rank measure there is a monotone boolean function computable in NP with $2^{\Theta(n)}$ lower bounds, where n is the number of input variables. All prior strongly exponential lower bounds for these models have been proved via the counting arguments discussed in Chapter 1; the lower bounds for other functions were, at best, weakly exponential (for monotone formulas, switching networks, and comparator circuits) or quasipolynomial (for monotone span programs).

It is important to note a subtlety here — one may suspect that we could use Theorem 5.0.1 to improve the separation in Theorem 6.4.4 to polynomial versus *strongly* exponential. However, the polynomial *upper bound* in Theorem 6.4.4 uses the *first* part of Theorem 5.0.1, which requires a gadget of rank (and thus size) n^2, whereas the lower bound uses a gadget of rank (and thus size) $2^{1/\varepsilon}$. Since we must use different gadgets to prove the upper and lower bounds, the instances of CSP-SAT we obtain are actually different, and so there is no separation!

Quantitatively, the best lower bounds one can hope for are of the form $2^{(1-o(1))n}$, whereas a closer analysis of our proof yields bounds of the form $2^{\alpha n}$ where α is an *extremely* small constant, on the order of 2^{-2000} or so. Ultimately, α is extremely small since our proof of the second part of Theorem 5.0.1 requires $\mathrm{rank}(g) \approx 2^{1/\varepsilon}$; improving this dependence of $\mathrm{rank}(g)$ on ε is an interesting open problem.

Chapter 7

Conclusion

In this thesis we have provided a unified framework in which it is possible to obtain nearly every prior lower bound for monotone circuit classes weaker than monotone circuits, as well as proving many new ones. In what remains, we would like to record several open problems related to this work that we find particularly intriguing.

Problem 1. Average-Case Monotone Circuit Lower Bounds. One significant limitation of our lower bound framework is that it only applies to *worst-case computation*. A natural question is whether or not one can obtain similar lower bounds in the *average-case* setting. In other words, suppose we have a "natural" probability distribution τ on $\{0,1\}^n$. Can we still prove our lower bounds even when the inputs are selected according to this distribution, and the corresponding monotone circuit models are allowed to make errors with probability ε? Many other circuit lower bounds in the literature can be extended in this way to several natural distributions (see e.g. [24,54,58]). This is especially well-motivated for the *uniform distribution*, as proving lower bounds on monotone circuit complexity against the uniform distribution is closely connected to *non-monotone circuit lower bounds* — such lower bounds were recently obtained by Rossman for monotone formulas computing a function closely related to st-connectivity [58].

Problem 2. Strongly Exponential Lower Bounds for Monotone Circuits. All of our lower bounds proceed by studying Razborov's rank measure $\mu_{\mathbf{F}}$. In Section 6.3 we showed weakly exponential lower bounds on the rank measure for GEN_n, which is a monotone function computable by polynomial-size monotone circuits. This implies our techniques can not prove lower bounds on monotone circuit complexity, leaving the following natural open problem: can we prove strongly exponential (i.e. $2^{\Theta(n)}$) lower bounds on the size of monotone circuits computing a monotone function in NP? The current strongest known lower bounds against such a boolean function is $2^{\Omega((n/\log n)^{1/3})}$, by Harnik and Raz [29].

Problem 3. Improving the Strongly Exponential Lower Bounds. In Theorem 6.4.5, for each prime p and each field $\mathbf{F}$ of characteristic $q \neq p$ we produce a monotone function computable in NP requiring strongly exponential $\mathbf{F}$-monotone span program size. An obvious downside of this result is that we do not have a *single* function computable in NP such that monotone span programs over *any field* require strongly exponential lower bounds. By Theorem 5.0.1, this would follow from resolving the following open problem: construct a bounded-width, linear-size unsatisfiable CNF $\mathcal{C}$ formula such that for *every* field $\mathbf{F}$ we have $\mathsf{NS}_{\mathbf{F}}(\mathcal{C}) = \Omega(n)$.

Furthermore, our construction only yields strongly exponential lower bounds for a function computable in NP. For a *truly* maximal separation, one could ask for a monotone function computable by *polynomial-size monotone circuits* (or, even polynomial-size *non-monotone* circuits) that requires strongly exponential rank measure — we do not even have a candidate function for either of these examples!

Problem 4. Razborov's Rank Measure vs. the Algebraic Tiling Number. In Theorem 4.3.1 we have shown that the algebraic gap complexity $\mathsf{gap}_{\mathbf{F}}(\mathcal{C})$ is exactly the same as the Nullstellensatz degree $\mathsf{NS}_{\mathbf{F}}(\mathcal{C})$. After we apply our lower-bounds framework, this implies that Razborov's rank measure $\mu_{\mathbf{F}}$ is "essentially" the same as the algebraic tiling number $\chi_{\mathbf{F}}$ for the monotone CSP-SAT functions $\mathsf{SAT}_{H,\Sigma}$. One could conjecture that this holds in general: namely, that $\mu_{\mathbf{F}}(f) = \chi_{\mathbf{F}}(f)$ for *every* boolean function f. However, we know that the connection *fails* for *non-monotone* functions. That is, if we consider the general (i.e. non-monotone) Karchmer-Wigderson games KW_f, Gál has shown that $\chi_{\mathbf{F}}(\mathsf{KW}_f)$ still captures *non-monotone* span program size [25]; but Razborov [53] has shown that the rank measure $\mu_{\mathbf{F}}(\mathsf{KW}_f)$ is always at most $O(n)$. This suggests the following open problem: find a natural monotone function such that $\chi_{\mathbf{F}}(f)$ is large, but $\mu_{\mathbf{F}}(f)$ is small.

Problem 5. "Operationalizing" Razborov's Rank Measure. When they introduced the Karchmer-Wigderson games, Karchmer and Wigderson [37] also showed that the *deterministic communication complexity* of KW_f exactly captures the formula size of f, and that the deterministic communication complexity of mKW_f captures the *monotone* formula size of f. Similarly, Gál [25] showed that the *algebraic tiling number* $\chi_{\mathbf{F}}$ captures span program size. In other work, Razborov [56] gave a complexity measure of KW_f (and mKW_f) which captures circuit size. Can one work "in reverse", and find a natural computational model which corresponds to the rank measure $\mu_{\mathbf{F}}$? Or, in other words, is this a general phenomena — is it the case that for *every* "natural" complexity measure of KW_f (or mKW_f) there is a corresponding "circuit" model?

Bibliography

[1] M. Ajtai, J. Komlos, and E. Szemeredi. An $O(n \log n)$ sorting network. *Proceedings of STOC 1983*, pages 1–9, 1983.

[2] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002.

[3] A. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl.*, 31:530–534, 1985.

[4] A. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π schemes. *Moscow University Mathematical Bulletin*, 41:63–66, 1987.

[5] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999.

[6] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on Hilbert's Nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806, 1994.

[7] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion, 1996.

[8] Amos Beimel. *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, chapter Secret-Sharing Schemes: A Survey, pages 11–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[9] Amos Beimel, Anna Gál, and Mike Paterson. Lower bounds for monotone span programs. *Computational Complexity*, 6(1):29–45, 1997.

[10] Amos Beimel and Enav Weinreb. Separating the power of monotone span programs over different fields. *SIAM J. Comput.*, 34(5):1196–1215, 2005.

[11] Ravi Boppana and Noga Alon. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):122, 1987.

[12] Allan Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6(4):733–744, 1977.

[13] Josh Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi. Homogenization and the polynomial calculus. *Computational Complexity*, 11(3-4):91–108, 2002.

[14] Samuel R. Buss. Lower bounds on Nullstellensatz proofs via designs. In *Proof Complexity and Feasible Arithmetics, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, April 21-24, 1996*, pages 59–72, 1996.

[15] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci.*, 62(2):267–289, 2001.

[16] Samuel R. Buss, Russell Impagliazzo, Jan Krajícek, Pavel Pudlák, Alexander A. Razborov, and Jirí Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1997.

[17] Samuel R. Buss and Toniann Pitassi. Good degree bounds on Nullstellensatz refutations of the induction principle. In *Proceedings of the Eleveth Annual IEEE Conference on Computational Complexity, Philadelphia, Pennsylvania, USA, May 24-27, 1996*, pages 233–242, 1996.

[18] Siu Man Chan and Aaron Potechin. Tight bounds for monotone switching networks via Fourier analysis. *Theory of Computing*, 10:389–419, 2014.

[19] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. *J. ACM*, 63(4):34:1–34:22, 2016.

[20] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971.

[21] Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 29–33, 1973.

[22] Stephen A. Cook, Yuval Filmus, and Dai Tri Man Le. The complexity of the comparator circuit value problem. *TOCT*, 6(4):15:1–15:44, 2014.

[23] László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Sci. Math. Hungary*, 32(3-4):429–437, 1996.

[24] Yuval Filmus, Toniann Pitassi, Robert Robere, and Stephen A. Cook. Average case lower bounds for monotone switching networks. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 598–607, 2013.

[25] Anna Gál. A characterization of span program size and improved lower bounds for monotone span programs. *Computational Complexity*, 10(4):277–296, 2001.

[26] Michael T. Goodrich. Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time. *Proceedings of STOC 2014*, pages 684–693, 2014.

[27] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 257–266, 2015.

[28] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 847–856, 2014.

[29] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 378–387. ACM, 2000.

[30] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986.

[31] Johan Håstad. The shrinkage exponent of De Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[32] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 233–248, 2012.

[33] Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993.

[34] Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, pages 353–364, 2002.

[35] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[36] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

[37] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[38] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993.

[39] Pravesh K. Kothari, Raghu Meka, and Prasad Raghavendra. Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 590–603, 2017.

[40] Jan Krajícek. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997.

[41] O.B. Lupanov. A method of circuit synthesis. *Izvesitya ZUV*, 1:120–140, 1958. (In Russian).

[42] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families IV: bipartite ramanujan graphs of all sizes. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1358–1377. IEEE Computer Society, 2015.

[43] E.I. Nechiporuk. On the complexity of schemes in some bases containing nontrivial elements with zero weights. *Problemy Kibernetiki*, 8:123–160, 1962.

[44] E.I. Nechiporuk. On a boolean function. *Soviet Math Dokl.*, 7(4):999–1000, 1966.

[45] Igor C. Oliveira. *Unconditional lower bounds in complexity theory*. PhD thesis, Columbia University, 2015.

[46] Mike Paterson and Uri Zwick. Shrinkage of de˜morgan formulae under restriction. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 324–333, 1991.

[47] Nicholas Pippenger. The complexity of monotone boolean functions. *Mathematical Systems Theory*, 11:289–316, 1978.

[48] Aaron Potechin. Bounds on monotone switching networks for directed connectivity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 553–562, 2010.

[49] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

[50] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 287–292, 1990.

[51] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.

[52] A. A. Razborov. Lower bounds on the size of switching-and-rectifier networks for symmetric boolean functions. *Mathematical Notes of the Academy of Sciences of the USSR*, 48(6):7991, 1990.

[53] A. A. Razborov. On submodular complexity measures. In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 76–83, New York, NY, USA, 1992. Cambridge University Press.

[54] Alexander A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Math. Dokl.*, 31:354–357, 1985.

[55] Alexander A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990.

[56] Alexander A. Razborov. Lower bounds for propositional proofs and independence results in bounded arithmetic. In *Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings*, pages 48–62, 1996.

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

[58] Benjamin Rossman. Correlation bounds against monotone NC^1. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 392–411. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[59] Claude Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949.

[60] Claude Shannon and John Riordan. The number of two-terminal series parallel networks. *Journal of Mathematical Physics*, 21:83–93, 1942.

[61] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[62] Phillip M Spira. On time-hardware complexity tradeoffs for boolean functions. In *Fourth Hawaii Symposium on System Sciences*, pages 525–527, 1971.

[63] B.A. Subbotovskaya. On realizations of linear functions by formulas using +, ., -. *Soviet Math Dokl.*, 2:323–336, 1961.

[64] A. Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Stanford University, 1990.

[65] Avi Wigderson. $\oplus\mathsf{L/poly} = \mathsf{NL/poly}$. `http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W94/proc.pdf`. Accessed: 2017-09-30.

[66] S.V. Yablonskii and V.P. Kozyrev. Mathematical problems of cybernetics. *Information Materials of Scientific Council of Akad. Nauk SSSR on Complex Problem "Kibernetika"*, 19a:3–15, 1968.