Some Hardness Escalation Results in Computational Complexity Theory

by

Pritish Kamath

B.Tech. Indian Institute of Technology Bombay (2012)
S.M. Massachusetts Institute of Technology (2015)

Submitted to Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering & Computer Science

at

Massachusetts Institute of Technology

February 2020

Author: ....
Department of Electrical Engineering and Computer Science
September 16, 2019

Certified by: ....
Ronitt Rubinfeld
Professor of Electrical Engineering and Computer Science, MIT
Thesis Supervisor

Certified by: ....
Madhu Sudan
Gordon McKay Professor of Computer Science, Harvard University
Thesis Supervisor

Accepted by: ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science, MIT
Chair, Department Committee on Graduate Students

Some Hardness Escalation Results in Computational Complexity Theory

by

Pritish Kamath

Submitted to Department of Electrical Engineering and Computer Science
on September 16, 2019, in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science & Engineering

Abstract

In this thesis, we prove new *hardness escalation* results in computational complexity theory; a phenomenon where hardness results against seemingly weak models of computation for any problem can be *lifted*, in a black box manner, to much stronger models of computation by considering a simple gadget composed version of the original problem.

For any unsatisfiable CNF formula F that is hard to refute in the *Resolution* proof system, we show that a gadget-composed version of F is hard to refute in any proof system whose lines are computed by efficient communication protocols. This allows us to prove new lower bounds for:

- **Monotone Circuit Size :** we get an exponential lower bound for an explicit monotone function computable by linear sized monotone span programs and also in (non-monotone) NC^2.
- **Real Monotone Circuit Size :** Our proof technique extends to *real* communication protocols, which yields similar lower bounds against *real* monotone circuits.
- **Cutting Planes Length :** we get exponential lower bound for an explicit CNF contradiction that is refutable with logarithmic Nullstellensatz degree.

Finally, we describe an intimate connection between computational models and communication complexity analogs of the sub-classes of TFNP, the class of all *total* search problems in NP. We show that the communication analog of PPA_p captures span programs over $\mathbb{F}_p$ for any prime p. This complements previously known results that communication FP captures formulas (Karchmer–Wigderson, 1988) and that communication PLS captures circuits (Razborov, 1995).

Thesis Supervisor: Ronitt Rubinfeld
Title: Professor of Electrical Engineering and Computer Science, MIT

Thesis Supervisor: Madhu Sudan
Title: Gordon McKay Professor of Computer Science, Harvard University

Acknowledgements

I have been very fortunate to be at MIT with great advisors, collaborators, friends and family.

I am extremely grateful to Madhu for taking me as his student, guiding me carefully and helping me navigate different kinds of situations. Madhu adapted his advising style for me over the years: very hands on initially, when I was figuring out research areas, but later, encouraging me to pursue directions on my own with other collaborators. I'm always amazed at Madhu's profound insights on virtually any topic; something I'll greatly miss after MIT.

I have been truly blessed to also have Ronitt as an advisor. Ronitt's guidance has been very valuable; she has provided me unconditional support and has been extremely caring and encouraging in whatever directions I wished to pursue.

The work in this thesis has been a result of guidance I found rather serendipitously. My initiation to the topics studied in this thesis happened when I started a reading group on query & communication complexity with Shalev Ben-David and Robin Kothari among others. Later, I had the opportunity to learn from Raghu Meka, who hosted me at UCLA on two occasions. Raghu's clear thinking and emphasis on foundational problems is something I aspire for. I'm also grateful to Raghu for serving on my committee. Finally, I had the fortune of working with Mika Göös, with whom all the work in the thesis was done (along with other collaborators). Despite working with Mika closely for so long, I haven't been able to fully absorb his graphic design skills, crisp writing style and bouldering techniques (I can barely complete a V2). I often try to compensate for this by using Mika's signature hyperlink style.

I would like to thank all the collaborators I have had the good fortune of working with while at MIT: Jayadev Acharya, Mohammad Bavarian, Arnab Bhattacharyya, Ankit Garg, Badih Ghazi, Mika Göös, Bernhard Haeupler, Elad Haramaty, Shen Li, Toniann Pitassi, Prasad Raghavendra, Ronald Rivest, Robert Robere, Ankit Shah, Julie Shah, Dmitry Sokolov, Katerina Sotiraki, Madhu Sudan, Ameya Velingker, Tom Watson and Manolis Zampetakis. A special shout-out to Badih, with whom I have had numerous fruitful collaborations. I'm yet to assimilate Badih's resourcefulness, coolness and optimism while facing hard research problems.

MIT is the place it is, primarily because of the amazing people around. I have gained tremendously by interacting with faculty at MIT and I would like to thank Scott Aaronson, Aleksander Mądry, Dana Moshkovitz and Vinod Vaikuntanathan. I would especially like to thank Costis Daskalakis for giving me advice on many occasions and also serving on my committee. Finally I'm very grateful to Yael Kalai for valuable guidance and being such an inspiring source of energy.

I have had excellent and diverse TA opportunities at MIT which have served as valuable learning experiences and I'm grateful to Dana Moshkovitz, David Karger, Dina Katabi and Piotr Indyk for providing me the same.

For getting me interested in theoretical research and trying to build some discipline into me, I would like to thank Supratik Chakraborty, Nutan Limaye and Neeraj Kayal. I would also like to

thank Csaba Szepesvári for a wonderful internship opportunity at Google DeepMind.

For making the ToC group such a warm environment for discussions (technically and otherwise) I would like to thank all my fellow students. There are too many people to thank, so I'll try to keep it brief: Michael Forbes, Mohammad Bavarian, Matt Coudron, Alan Guo, Henry Yuen, John Wright, Robin Kothari, Shalev Ben-David, Badih Ghazi, Gautam "G" Kamath, Maryam Aliakbarpour, Jerry Li, Mădălina Persu, Luke Schaeffer, Adam Sealfon, Katerina Sotiraki, Prashant Vasudevan, Prabhanjan Ananth, Akshay Degwekar, Nishanth Dikkala, Govind Ramnarayan, Manolis Zampetakis, Sitan Chen … alas, that wasn't brief and I have still missed so many people. Additionally I would like to thank the admins Joanne, Rebecca and Debbie for cutting through all the bureaucracy with a pedestrian sense of ease, making ToC among the most fun groups at MIT! For making my visits to Harvard fun and enjoyable I would like to thank Mitali Bafna, Elad Haramaty and Preetum Nakkiran.

For making MIT a home, I would like thank Yashovardhan Chati and Devendra Shelar for tolerating me as a roommate (and all the fun alley cricket at home, with apologies to people living downstairs!) and Vaibhav, Divya, Krithika, Rajan, Saurabh, Shilpa, Ranjitha and Parnika for the never ending conversations on life, universe and everything.

Finally, I would like to thank my family who have raised me up to where I am today. This includes my parents, mom Urmila and aunt Naina, my brother Sudeep and sister-in-law Swarna (and a yet-to-arrive young one!). Sudeep has been a constant source of inspiration ever since I was tiny; he taught me all the important things in life very early on (including topics like combinatorics, trigonometry and calculus). Finally, I would like to thank my wife Apoorvaa, who brought along with her very inspiring in-laws Meenal and Chandrahas. I was incredibly lucky to have met Apoorvaa very early in life and I cannot imagine how life would have been otherwise.

Dedicated to the memory of my father, Uday Kamath,
who inspired the mathematical spark in both of his sons.

Contents

Chapter 1

Introduction

Understanding *computational hardness* of problems has been a central theme in computer science, arguably since the genesis of the field [Chu36, Tur37, Coo71, Kar72]; e.g. the P vs. NP question has been recognized as a Millenium Prize Problem by the Clay Mathematical Institute [Jaf06]. While understanding hardness results are philosophically important, they are also relevant practically as the security of most cryptographic systems are based on the computational hardness of specific problems. We seem very far from resolving the fundamental questions in complexity theory; nevertheless the field has made progress on many fronts such as proving some weak complexity separations (e.g. $\mathsf{NQP} \not\subseteq \mathsf{ACC}$ [Wil11, MW18]) and explicit lower bounds in restricted models (e.g. against AC^0 [Hås86, HRST17], $\mathsf{AC}^0[p]$ [Raz89, Smo87], monotone circuits [Raz85b, AB87, Tar88], etc.). The reader is highly encouraged to read the wonderful book by Avi Wigderson [Wig19] to get an overview of Computational Complexity Theory and its relevance to Computer Science, Mathematics and beyond.

In this thesis, we develop new *hardness escalation* results; a phenomenon where hardness results against seemingly weak models of computation for any problem can be *lifted*, in a black box manner, to much stronger models of computation by considering a simple gadget composed version of the original problem. We prove new lower bounds against a range of monotone computational models and propositional proof systems. At the heart of our proofs, we exploit connections to query and communication complexity and our hardness escalation results fall under an emerging paradigm, that has come to be known as ***query-to-communication lifting***.

1.1 Complexity Theory

We briefly describe the different models of complexity theory that we study in this thesis. A more comprehensive background is provided in Chapter 2.

Monotone Computational Models. *Boolean monotone circuits* compute on inputs by iteratively performing AND and OR operations. *Real monotone circuits* [HC99, Pud97] generalize monotone circuits by having real valued intermediate computations, where the gates compute arbitrary monotone functions on its inputs. *Monotone span programs* [KW93] compute by testing whether a subspace indicated by the input spans a certain target vector.

Propositional Proof Systems. The goal of propositional proof systems is to prove unsatisfiability of CNFs by deriving a contradiction starting from the given clauses. The *Resolution* proof system operates via the resolution rule : $(A \vee x), (B \vee \neg x) \models (A \vee B)$ to derive new clauses until a contradiction is reached. The *Cutting Planes* proof system is a geometric generalization of Resolution, where intermediate statements are linear threshold functions. The $\mathbb{F}$-*Nullstellensatz* proof system (for any field $\mathbb{F}$) proves the unsatisfiability of a polynomial system $\{p_1 = 0, \ldots, p_m = 0\}$ by providing a set of polynomials $q_1, \ldots, q_m$ such that $\sum_i p_i q_i = 1$ (syntactically over $\mathbb{F}$).

Query Complexity. A *decision tree* algorithm solves a search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ on an unknown input $x \in \{0,1\}^n$ by repeatedly querying input variables. The goal is to find an answer $o \in \mathcal{O}$ such that $(x, o) \in S$, while minimizing the number of bits queried. This basic template can be instantiated in different types of computation (e.g. deterministic, randomized, nondeterministic). In this thesis, we will invoke close ties between query complexity and proof systems.

Communication Complexity. A *communication protocol* solves a search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ on inputs (x, y) that is shared between two players, by exchange of bits between the players. The goal is to find an answer $o \in \mathcal{O}$ such that $(x, y, o) \in S$, while minimizing the number of bits exchanged. Once again, this basic template can be instantiated for different types of computation. In this thesis, we will invoke close ties between communication complexity and monotone computational models (cf. [KW88]).

1.2 Outline of the thesis

In Chapter 2, we provide an introduction to basic models of computational models, proof systems and query & communication complexity that are relevant for this thesis. In Chapter 3, we recall the relevant background from some recent works on query-to-communication lifting theorems.

Chapter 4. Dag-like Models and Equivalences

Dag-like communication protocols [Raz95, Pud10, Sok17], generalize the usual notion of (tree-like) communication protocols, and captures circuit *size*, analogous to an equivalence between formulas and tree-like protocols [KW88]. In particular, the (monotone) circuit complexity of f is equal to the least size of a dag-like protocol that solves the (monotone) Karchmer–Wigderson search problem associated with f. *Real* dag-like communication protocols further generalize the notion of dag-like protocols. and in turn capture the real monotone circuit size [HP18].

Analogous to decision trees, we define *decision dags* (or conjunction-dags), which are equivalent to resolution refutations. Moreover, we also define *real* decision dags (or threshold-dags), which are equivalent to cutting planes refutations.

Chapter 5. Monotone Circuit Lower Bounds

In this chapter, we prove a *query-to-communication lifting theorem* that escalates lower bounds for decision dags to lower bounds for dag-like communication protocols, thereby yielding monotone circuit lower bounds. The result can be interpreted as a ***converse*** to *monotone feasible interpolation* [Kra97, BPR97], which is a popular method to prove Resolution refutation lower bounds for proof systems by reduction to monotone circuit lower bounds. A theorem of this type was conjectured by Beame, Huynh and Pitassi [BHP10].

We introduce a natural *monotone* encoding of constraint satisfaction problems (CSP). An illustrative example is the monotone encoding of 3Xor-Sat, for which we get an exponential monotone circuit size lower bound. This result stands in contrast to the fact that 3Xor-Sat is computable by a linear-size monotone $\mathbb{F}_2$-span program and hence is also computable in (non-monotone) NC^2, as there are fast parallel (non-monotone) algorithms for linear algebra [Mul87].

Chapter 6. Cutting Planes Lower Bounds

We significantly strengthen the lifting theorem from the previous chapter to *real* dag-like protocols. This allows us to get the same qualitative lower bounds against real monotone circuit size, as were obtained for monotone circuits.

Another application that follows are lower bounds for the (semantic) Cutting Planes proof system (similar in spirit to monotone feasible interpolation [Kra97]). In particular, for any unsatisfiable n-variate k-CNF F which requires Resolution *width* w, we get that a related unsatisfiable $n^{O(1)}$-variate $3k$-CNF F' requires Cutting Planes refutations of length $n^{\Omega(w)}$. By instantiating F carefully, this yields an unsatisfiable CNF that has $O(\log n)$-degree Nullstellensatz refutations, but requires Cutting Planes refutations of length $2^{n^{\Omega(1)}}$.

Chapter 7. Monotone Span Program Lower Bounds

We construct an explicit CNF (via reduction from [BR98]) that has $\mathbb{R}$-Nullstellensatz refutations of $O(1)$-degree, but for any prime p, requires $\mathbb{F}_p$-Nullstellensatz refutations of degree $n^{\Omega(1)}$.

We combine this with the lifting theorem from Nullstellensatz to Monotone Span programs [PR18] to get an explict function that has linear-sized monotone span programs over $\mathbb{R}$, but requires exponential sized monotone span program over $\mathbb{F}_p$ for any prime p.

Chapter 8. TFNP in Query & Communication Complexity

We describe an intimate connection between the models studied in this thesis and query & communication analogs of TFNP, the complexity class of all *total search problems in* NP.

Communication TFNP*.* Tree-like protocols are the communication analog of FP, the complexity class of all search problems solvable in polynomial time on Turing machines. Karchmer and Wigderson [KW88] characterized (monotone) formulas in terms of communication protocols. Dag-like communication protocols were introduced by Razborov [Raz95] as the communication analog of the TFNP-subclass PLS, and he showed that they characterize circuits. We contribute a third such characterization: We show that the communication analog of PPA captures $\mathbb{F}_2$–span programs. More generally, we show that communication PPA_p captures $\mathbb{F}_p$–span programs.

Query TFNP. Decision trees are the query analog of FP and they are known to characterize the depth complexity of resolution refutations [LNNW95]. Additionally, decision-dags capture the *width* complexity of resolution refutations [Pud00, AD08] and we show how to view this as a query analog of PLS. Again, we contribute more such characterizations: the query analog of PPA (resp. PPA_p) captures the degree of Nullstellensatz refutation over $\mathbb{F}_2$ (resp. $\mathbb{F}_p$).

Chapter 9 Summary and Open Problems

We highlight some key open problems and future directions in the context of this thesis.

1.3 Note on the content of this thesis

All the work in this thesis is based on prior publications [GGKS18, GKRS19] with co-authors Ankit Garg, Mika Göös, Robert Robere and Dmitry Sokolov. During the course of Ph.D., the author has worked on a variety of topics most of which have not been included as they are unrelated to the theme of this thesis. We briefly mention these lines of work here.

More Complexity & Information Theory

▷ [GKS16a] : we study the class of permutation-invariant communication problems, and show that the information complexity and randomized/deterministic communication complexity of all such functions are polynomially related (upto additive log factors).

▷ [GHKS17] : we study a setting of message compression in a distributed setting, where the players don't know the true message distribution and can learn it only through observed samples.

▷ [GKS16b, GKR18] : we consider the problem of non-interactive simulation studied in information theory. We show that the problem of checking whether a given joint distribution can be non-interactively simulated from another is decidable. A new technique of *dimension reduction for polynomials* is introduced (generalizing the Johnson-Lindenstrauss lemma).

▷ [BGH$^+$16] : we prove that the well-known protocol for correlated sampling is actually optimal.

▷ [GKPW19] : we prove a query-to-communication lifting theorem for the class P^{NP}.

▷ [GKSZ19] : In an ongoing work, we study the class PPA_q in detail and provide a natural complete problem for it.

Miscellaneous

▷ [HKV15] : we study the model of (noisy) communication with partial noiseless feedback and prove lower bounds on achievable rates in deterministic and randomized settings.

▷ [ABK17] : we study the problem of one-bit compressive sensing, providing improved upper and lower bounds on the number of measurements required.

▷ [SKLS18] : in the paradigm of *learning from demonstrations*, we use a Bayesian inference approach for infering linear temporal logic (LTL) formulas underlying demonstrations. This project was implemented in WebPPL, a probabilistic programming language.

Chapter 2

Complexity Theory Basics

In this chapter, we give an introduction to computational models and proof systems that we study in this thesis. We also cover some basics of query and communication complexity. This chapter is independent of the rest of the thesis and hopes to serve as an introductory reference. For more details, we refer the reader to the wonderful texts by Jukna [Juk12] and Krajícek [Kra19].

2.1 Computational Models

Circuits & Formulas. A circuit (aka straight-line program) is a basic model of computation that may be thought of as a "hardware implementation" of an algorithm for a fixed input size. A circuit is represented as a directed acyclic graph, where there are (i) n nodes of zero in-degree called input gates (ii) m nodes of zero out-degree called output gates and (iii) all non-input gates g are labeled by an operation Φ_g in the chosen *basis*. The circuit is said to compute a function $f : \{0,1\}^n \to \{0,1\}^m$, such that on a given input $z \in \{0,1\}^n$ corresponding to the n input gates, each non-input gate g computes the value $g(z) = \Phi_g(g_1(z), g_2(z), \ldots, g_t(z))$ where $g_1(z), \ldots, g_t(z)$ are values computed by the predecessor gates of g, and finally, $f(z)$ corresponds to the bits computed by the m output gates. The *size* of a circuit is the total number of gates[1] and its *depth* is the length of the longest path from an input to an output gate. A formula is a special case of a circuit, where the directed acyclic graph is a tree.

[1] often, *size* is also defined as the number of "wires", but these measures are at most quadratically apart.

Boolean Circuits & Formulas. A Boolean circuit has operations that belong to the *de Morgan* basis $\{\wedge, \vee, \neg\}$ consisting of the logical AND, OR and NOT operations. It is well known that any algorithm that, on inputs of length n, runs in time $t(n)$ on a Turing machine can also be simulated by a Boolean circuit of size $O(t(n) \log t(n))$ (cf. [AB09]). Thus, Boolean circuits serve as a simple abstraction for potentially understanding computational hardness results.[2] The class $\mathsf{P/poly}$ is the set of languages computable by poly-sized family of Boolean circuits, whereas the class $\mathsf{NC}^1\mathsf{/poly}$ is the set of all languages computable by poly-sized Boolean formulas. For ease of notation, we assume circuits and formulas to be Boolean by default.

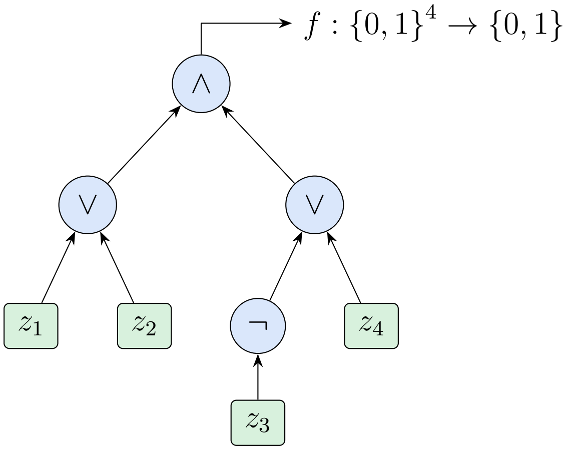


Figure 2.1: Example of a (non-monotone) Boolean formula

Definition 2.1 (Monotone Function)**.** A function $f : \{0,1\}^n \to \{0,1\}$ is said to be *monotone* if $f(x) \leq f(y)$ for all $x, y \in \{0,1\}^n$ satisfying $x \preceq y$ (i.e. x is bit-wise dominated by y).

A Boolean circuit/formula is *monotone* if the gates are labeled by operations in $\{\wedge, \vee\}$, i.e. without NOT gates. The set of all monotone circuits precisely compute the set of all monotone functions.

Remark 2.2 (Fan-in of Boolean circuits)**.** While a Boolean circuit is allowed to have unbounded fan-in, we can always transform them to have fan-in at most 2, by blowing up the size of the circuit by at most a polynomial factor. In this thesis, we will only consider Boolean circuits with fan-in at most 2.

[2] The advent of artificial neural networks makes one wonder whether it serves more than just this purpose.

Monotone Real Circuits. A monotone real circuit [HC99, Pud97] of fan-in 2 has gates g labeled by operations $\Phi_g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that can be *arbitrary* monotone functions over the reals, that is, $\Phi_g(x_1, x_2) \leq \Phi_g(y_1, y_2)$ whenever $x_1 \leq y_1$ and $x_2 \leq y_2$ (see [Juk12, §9.6] for exposition). While the intermediate gates are allowed to compute (monotone) functions $g : \{0,1\}^n \to \mathbb{R}$, we require that the final output is still Boolean.

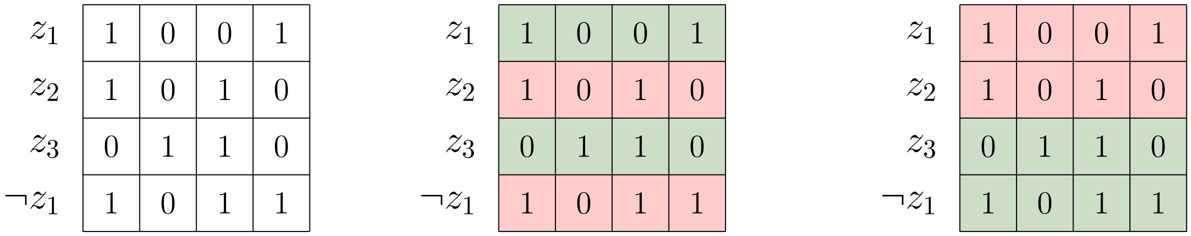
It is well known that monotone real circuits can be exponentially more powerful than even non-monotone Boolean circuits for computing certain functions [Ros97]![3] The original motivation to study such circuits, and what interests us in this thesis, is that lower bounds for monotone real circuits imply lower bounds for the *Cutting Planes* proof system [CCT87] (defined shortly).

Remark 2.3 (Fan-in of monotone real circuits)**.** Monotone real circuits can also be defined with fan-in k (instead of 2). However unlike Boolean circuits, fan-in reduction for monotone real circuits is not trivial! In fact, it was only recently shown that any monotone real circuit of size s and fan-in k can be simulated by a monotone real circuit of size $O(s \cdot n^{k-2})$ and fan-in 2 [HP18] (an exponential dependence in k is clearly necessary!).

Span Programs. Span programs are a model of computation introduced by Karchmer and Wigderson [KW93] (see also [Juk12, §8] for exposition), which are very different than circuits (or straight-line programs). For any field $\mathbb{F}$, an $\mathbb{F}$-*span program* is specified by a matrix $M \in \mathbb{F}^{m \times m'}$, each row of which is labeled by a literal, z_i or $\neg z_i$. We say that the program accepts an input $z \in \{0,1\}^n$ iff the rows of M whose labels are consistent with z (literals evaluating to TRUE on z) span the all-1 row vector (see Figure 2.2 for an illustration). The *size* of a span program is its number of rows m (it is known that without loss of generality, one can take $m' \leq m$).

A span program is *monotone* if all its literals are positive; in this case the program computes a monotone function. Note that a monotone span program computes only monotone functions because including more rows can only help in spanning the all-1 row vector. In fact, all monotone functions are computable by monotone span programs. An interesting property of monotone span programs is that they are exactly equivalent to *linear secret sharing schemes* in Cryptography.

[3] another neat observation is that monotone real circuits can simulate all neural networks with non-negative weights and monotone activations (such as ReLU, sigmoid and tanh).



z_1	1	0	0	1
z_2	1	0	1	0
z_3	0	1	1	0
$\neg z_1$	1	0	1	1

(a) Span Program

z_1	1	0	0	1
z_2	1	0	1	0
z_3	0	1	1	0
$\neg z_1$	1	0	1	1

(b) Accepts on input $(1, 0, 1)$

z_1	1	0	0	1
z_2	1	0	1	0
z_3	0	1	1	0
$\neg z_1$	1	0	1	1

(c) Rejects on input $(0, 0, 1)$

Figure 2.2: Example of (non-monotone) $\mathbb{F}_2$-Span Program

2.2 Propositional Proof Complexity

Propositional proof systems operate on Boolean formulas, the simplest case being CNF formulas, which is a conjunction (And) of clauses, where each clause is a disjunction (Or) of literals, e.g. $F = (z_1 \vee z_2) \wedge (\neg z_1 \vee z_2) \wedge (\neg z_2)$. A CNF formula is said to unsatisfiable if no assignment to its variables satisfies the formula.

A proof of the unsatisfiability of F starts with clauses of F (called axioms), applies several simple (fixed set of) rules to produce a contradiction. The main goal of proof complexity is to show that some unsatisfiable CNF formulas require long/complex proofs — it is well known that $\mathsf{NP} = \mathsf{coNP}$ iff there is a propositional proof system giving rise to short (polynomial in size of F) proofs of unsatisfiability of all unsatisfiable CNF formulas F [CR79].

Resolution. The Resolution proof system, introduced by Blake [Bla38], has become popular as a theorem-proving procedure, e.g. DPLL due to Davis and Putnam [DP60], Robinson [Rob65] (see also [Juk12, §18] for exposition). Let F be any unsatisfiable CNF. A *Resolution refutation* of F is a sequence of clauses $\mathcal{C} = (C_1, \ldots, C_t)$ where $C_t = \perp$ is the empty clause and every C_i is either a clause of F or is derived from some previous two clauses using the *resolution rule*:

$$\frac{A \vee z_i \qquad B \vee \neg z_i}{A \vee B}$$

meaning that the clause $A \vee B$ can be inferred from the clauses $A \vee z_i$ and $B \vee \neg z_i$. We say that the variable z_i was *resolved* to derive the clause $A \vee B$. The *length* of the proof is equal to the total number of clauses in the proof. The *width* of the proof is the maximum number of literals present

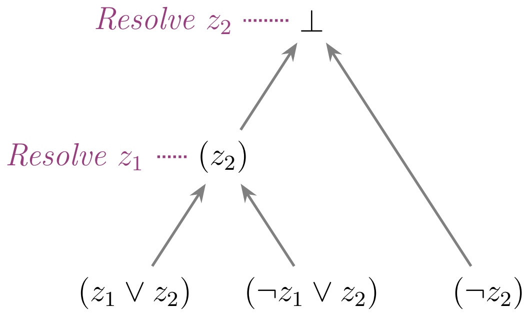


Figure 2.3: Example of a (tree-like) resolution refutation

in any intermediate clause. The *resolution width* of F is the least width of any resolution refutation of F. A proof is said to be *tree-like* if the sequence of derivations form a tree as opposed to a more general directed acyclic graph.

It is well known that the Resolution proof system is *sound* and *complete*, i.e. a CNF is unsatisfiable iff it is has a resolution refutation.

Cutting Planes. The Cutting Planes proof system originated in the works on integer programming by Gomory [Gom63] and Chvátal [Chv73] and considered as a proof system by Cook, Coullard and Turán [CCT87]. It is more powerful than the Resolution proof system and can be viewed as a "geometric generalization" (see [Juk12, §19] for exposition). Given a matrix $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, we say that the system $Az \geq b$ is unsatisfiable if it has no feasible point $z \in \{0,1\}^n$. A *Cutting Planes refutation* of $Az \geq b$ is a sequence of linear inequalities $\mathcal{L} = (L_1, \ldots, L_t)$ where L_t is $0 \geq 1$ and every L_i is either an inequality in $Az \geq b$, or an inequality $z_i \geq 0$ or $-z_i \geq -1$ or is derived from some previous inequalities via one of the following rules (where $a \in \mathbb{Z}^n$, $b, c \in \mathbb{Z}$ and $c \geq 1$):

$$\frac{a^\top z \geq b}{c \cdot a^\top z \geq c \cdot b} \qquad \frac{a_1^\top z \geq b_1 \qquad a_2^\top z \geq b_2}{(a_1 + a_2)^\top z \geq b_1 + b_2} \qquad \frac{c \cdot a^\top z \geq b}{a^\top z \geq \lceil b/c \rceil}$$

The *length* of the proof is equal to the total number of inequalities in the proof.

In order to refute unsatisfiable CNFs, we can represent a CNF F as a system of linear inequalities $A_F z \geq b_F$. This is done by representing each clause by an inequality using the translation $z_i \rightsquigarrow z_i$ and $\neg z_i \rightsquigarrow (1 - z_i)$. For example, $z_1 \vee z_2 \vee \neg z_3$ corresponds to the inequality $z_1 + z_2 - z_3 \geq 0$

(equivalent to $z_1 + z_2 + (1 - z_3) \geq 1$). It is easy to see that Cutting Planes can simulate Resolution as demonstrated by the following example,

$$\frac{z_1 \vee z_2 \vee \neg z_3 \qquad z_1 \vee z_2 \vee z_3}{z_1 \vee z_2} \quad \rightsquigarrow \quad \frac{z_1 + z_2 - z_3 \geq 0 \qquad z_1 + z_2 + z_3 \geq 1}{2z_1 + 2z_2 \geq 1}$$

$$\text{and} \quad \frac{2z_1 + 2z_2 \geq 1}{z_1 + z_2 \geq 1}$$

While the rules we described above allow us to *syntactically* derive new inequalities from earlier ones, a stronger version that is often studied is the *semantic* Cutting Planes proof system. Here, we are allowed "derive" an inequality L_k from previous inequalities L_i and L_j as long as $L_k(z)$ holds only if both $L_i(z)$ and $L_j(z)$ hold, for all $z \in \{0,1\}^n$. The restriction on z being in $\{0,1\}^n$, makes it coNP-hard to decide if L_k can be semantically derived from L_i and L_j (by a reduction from the knapsack problem). While semantic Cutting Planes refutation is significantly stronger than the syntactic version, some proof techniques (including those in this thesis) in fact give lower bounds on the length of semantic Cutting Planes refutations.

Nullstellensatz. Let $P \coloneqq \{p_1 = 0, p_2 = 0, \ldots, p_m = 0\}$ be an unsatisfiable system of polynomial equations in $\mathbb{F}[z_1, z_2, \ldots, z_n]$ for a field $\mathbb{F}$. An $\mathbb{F}$-*Nullstellensatz refutation* of P is a sequence of polynomials $q_1, q_2, \ldots, q_m \in \mathbb{F}[z_1, z_2, \ldots, z_n]$ such that $\sum_{i=1}^{m} q_i p_i = 1$, where the equality is syntactic. The *degree* of the refutation is $\max_i \deg(q_i p_i)$. The $\mathbb{F}$-*Nullstellensatz degree* of P, denoted by $\mathsf{NS}_{\mathbb{F}}(P)$, is the least degree of an $\mathbb{F}$-Nullstellensatz refutation of P. See [Kra19, §16.1] for exposition.

In order to refute unsatisfiable CNFs, we can represent a CNF F as a polynomial system P_F. This is done by representing each clause containing k literals as a degree k polynomial using the translation $z_i \rightsquigarrow (1 - z_i)$ and $\neg z_i \rightsquigarrow z_i$. For example, $(z_1 \vee \neg z_2 \vee \neg z_3)$ corresponds to the polynomial $(1 - z_1) z_2 z_3 = 0$. In the case of $\mathbb{F} \neq \mathbb{F}_2$, we also include the *Boolean axioms* $z_i^2 - z_i = 0$. For ease of notation, we use $\mathsf{NS}_{\mathbb{F}}(F)$ to denote $\mathsf{NS}_{\mathbb{F}}(P_F)$.

Example. Let $F = (z_1 \vee z_2) \wedge (\neg z_1 \vee z_2) \wedge (\neg z_2)$ encoded as a polynomial system $P = \{p_1 \coloneqq (1 - z_1)(1 - z_2) = 0,\ p_2 \coloneqq z_1(1 - z_2) = 0,\ p_3 \coloneqq z_2 = 0\}$. A $\mathbb{F}$-Nullstellensatz refutation (for any $\mathbb{F}$) of P with degree 2 is simply $q_1 \coloneqq 1$, $q_2 \coloneqq 1$ and $q_3 \coloneqq 1$, since, $p_1 q_1 + p_2 q_2 + p_3 q_2 = 1$.

Remark 2.4 (CNF vs CSP)**.** We would often talk about refuting a constraint satisfaction problem (CSP), where the constraints can come from an general set of predicates. The implicit understanding would be that we refute the natural CNF-encoding of the CSP. Eg. $z_1 \oplus z_2 \oplus z_3 = 0$ can be encoded as $(\neg z_1 \vee \neg z_2 \vee \neg z_3) \wedge (\neg z_1 \vee z_2 \vee z_3) \wedge (z_1 \vee \neg z_2 \vee z_3) \wedge (z_1 \vee z_2 \vee \neg z_3)$.

2.3 Query Complexity

The basic objects of study in query complexity are *decision trees*. For an n-bit boolean function $f : \{0,1\}^n \to \{0,1\}$, a decision tree algorithm on an unknown input $z \in \{0,1\}^n$, repeatedly queries individual input variables. In each step, the algorithm specifies a coordinate $i \in [n]$ and gets to learn $z_i \in \{0,1\}$. The key question is: *How many queries are needed to evaluate f?*

This basic template can be instantiated for many different types of computation such as deterministic, nondeterministic, randomized, etc. While the holy grail of complexity theory is to prove separations between different Turing machine based computational models (e.g., $\mathsf{P} \neq \mathsf{NP}$), these questions remain hopelessly out of reach. Query complexity originated in the context of separating complexity classes relative to oracles, starting from the work of Baker, Gill and Solovay [BGS75]; see Vereshchagin [Ver99] for an exposition. Buhrman and de Wolf [BdW02] and Jukna [Juk12] have excellent surveys on query complexity.

In this thesis, we will primarily consider *search problems* $S \subseteq \{0,1\}^n \times \mathcal{O}$ (also known as *relational problems*). Here the goal of a decision tree algorithm is to output any $o \in S(z) := \{o \in \mathcal{O} : (z,o) \in S\}$ if it exists or $\perp$ if $S(z) = \emptyset$. In fact, we will only consider *total search problems*, where $S(z) \neq \emptyset$ for all $z \in \mathcal{Z}$. For any search problem S, we denote it's deterministic decision tree complexity by $\mathsf{FP}^{\mathsf{dt}}(S)$. We use this (rather unusual) notation to draw a parallel between deterministic query complexity and the Turing machine complexity class FP — this analogy is elaborated on in Chapter 8.

CSP search problems. In this thesis, we will exploit an intimate connection between query complexity and proof complexity for which, we will consider a canonical search problem associated to any unsatisfiable n-variate CNF. More generally, we define it for any CSP.

Definition 2.5 (CSP Search Problems)**.** For any unsatisfiable CSP $F = \bigwedge_{i \in [m]} D_i$, the CSP search problem $S(F) \subseteq \{0,1\}^n \times [m]$ is defined as,

Input: an n-variate truth assignment $z \in \{0,1\}^n$

Output: $i \in [m]$ such that constraint D_i of F is falsified by z (i.e., $D_i(z) = 0$)

2.4 Communication Complexity

In the basic model of communication complexity, introduced by Yao [Yao79], two players Akbar and Birbal[4] share the input to a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where Akbar holds $x \in \{0,1\}^n$ and Birbal holds $y \in \{0,1\}^n$. The wish to compute the value $f(x,y)$ by communicating with each other. The key question is: *How many bits of communication are needed to compute f?*

As with query complexity, this basic template can be instantiated for many different types of computation such as deterministic, nondeterministic, randomized, etc. Despite it's simplicity, communication complexity is reputed for being vastly powerful in the study of many diverse areas within complexity theory. Quoting Avi Wigderson [Wig19, Chapter 15], "*communication is an important computational resource in distributed systems – but in [several] applications through simple or subtle reductions it informs us about other computational resources like time, space, size, randomness, queries, chip area and more*". Several textbooks have been dedicated to the study of communication complexity and its applications [KN97, LS09, RY17].

In this thesis, as with query complexity, we will primarily consider *search problems* $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$. Here, Akbar holds $x \in \mathcal{X}$, Birbal holds $y \in \mathcal{Y}$ and they wish to output any $o \in S(x,y) := \{o \in \mathcal{O} : (x,y,o) \in S\}$ if it exists or $\perp$ if $S(x,y) = \emptyset$. As in query complexity, we will only consider *total search problems*, where $S(x,y) \neq \emptyset$ for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$. For any search problem S, we denote it's deterministic communication complexity by $\mathsf{FP}^{\mathsf{cc}}(S)$, drawing an analogy with the Turing machine complexity class FP.

[4] Akbar was the third Mughal emperor (1556-1605) of India. Birbal was Akbar's advisor and main commander of the army. Akbar & Birbal are part of many Indian folk tales, which focus primarily on Birbal's sharp wit.

Karchmer-Wigderson search problems. In a seminal work, Karchmer and Wigderson [KW88] established an equivalence between the (monotone) circuit-depth complexity of a (monotone) function $f : \{0,1\}^n \to \{0,1\}$ and the communication complexity of the so-called *(monotone) Karchmer–Wigderson search problem* defined as follows.

Definition 2.6 (Karchmer-Wigderson search problem)**.** For any (monotone) function $f : \{0,1\}^n \to \{0,1\}$, the search problem $\mathsf{KW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [n]$ (resp. $\mathsf{mKW}(f)$) is defined as,

$$\begin{aligned} \textit{Input:} \quad & (x,y) \in f^{-1}(1) \times f^{-1}(0) \\ \textit{Output:} \quad & i \in [n] \text{ such that } x_i \neq y_i \text{ (resp. } x_i > y_i) \end{aligned}$$

Theorem 2.7 ([KW88])**.** *For any function $f : \{0,1\}^n \to \{0,1\}$,*

$$\begin{aligned} \mathsf{FP}^{\mathsf{cc}}(\mathsf{KW}(f)) &= \textit{circuit depth of } f \\ \mathsf{FP}^{\mathsf{cc}}(\mathsf{mKW}(f)) &= \textit{monotone circuit depth of } f \ \textit{(for monotone } f\textit{)} \end{aligned}$$

This characterization was used to prove an $\Omega(\log^2 n)$ lower bound on the circuit depth of the monotone function s-t-Connectivity, the problem of deciding if a given graph (as an adjacency matrix) has a path between designated nodes s and t. This implies an $n^{\Omega(\log n)}$ lower bound on the size of monotone formulas computing s-t-Connectivity, due to the following lemma.

Lemma 2.8. *For any $f : \{0,1\}^n \to \{0,1\}$, the (monotone) circuit depth of f is equal, upto constant factors, to the log of the (monotone) formula size of f.*

Theorem 2.7 has had a tremendous influence in the area of proving formula lower bounds. Karchmer, Raz and Wigderson [KRW95] formulated a direct-sum type conjecture in communication complexity, which implies $\mathsf{P} \not\subseteq \mathsf{NC}^1$ — this conjecture has come to be known as the KRW conjecture. While this still remains open, the best known (non-monotone) formula lower bound of $n^{3-o(1)}$ has been obtained via this approach [DM18], although there have been alternative approaches as well.

There has been much more progress on the front of proving *monotone* formula size lower bounds. Raz and Wigderson [RW92] showed that the Perfect-Matching function on a graph of size n requires monotone formulas of size $2^{\Omega(n)}$ (note: this is $2^{\Omega(\sqrt{N})}$ where $N := \binom{n}{2}$ is the number of variables). A fundamental result, very relevant to this thesis, is due to Raz and McKenzie [RM99], who showed that for all $i \geq 1$, monotone-$\mathsf{NC}^{i+1} \not\subseteq$ monotone-NC^{i}. More importantly, this result kick-started the paradigm of *query-to-communication* lifting that we further explore in this work. By now, near-maximal lower bounds are known on the monotone formula size of explicit functions: Göös and Pitassi [GP18b] showed a lower bound of $2^{\Omega(n/\log n)}$ for an explicit function in NP. A lower bound of $2^{\Omega(n)}$ can be recovered from the result of Pitassi and Robere [PR18] for an explicit function in P and a lower bound of $2^{\Omega(n/\text{polylog } n)}$ can be recovered from an upcoming result of de Rezende et al. [dRMN+19] for an explicit function in monotone-P.

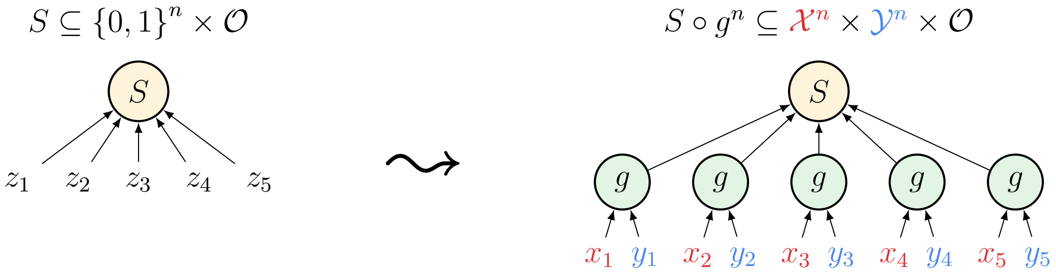
2.5 Query-to-Communication Lifting

It is easy to see that communication protocols are at least as powerful as decision trees (where, we assume the input $\{0,1\}^n$ is partitioned between Akbar and Birbal according to a fixed partition). A decision tree that solves a search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ using at most d queries can be simulated by a protocol that communicates at most d bits: a query to the i-th coordinate is simulated by having the player holding x_i send that bit to the other player.

The goal of a *lifting theorem* is to prove a converse. The template of a lifting theorem uses the following two-step strategy for proving lower bounds against communication protocols:

Step 1: Prove a *simulation theorem* showing that for a large class of communication problems $\widetilde{S}$, any communication protocol for $\widetilde{S}$ can be efficiently simulated by a decision tree solving a related problem S.

Step 2: Show that decision trees solving S require high cost.



$S \subseteq \{0,1\}^n \times \mathcal{O}$
S
z_1
z_2
z_3
z_4
z_5
$\rightsquigarrow$
$S \circ g^n \subseteq \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{O}$
S
g
x_1 y_1
x_2 y_2
x_3 y_3
x_4 y_4
x_5 y_5

Figure 2.4: Composed search problem $S \circ g^n$, where $S \subseteq \{0,1\}^n \times \mathcal{O}$ is an arbitrary n-bit search problem and $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is a carefully chosen two-party gadget.

In particular, we consider the family of *composed* (or *lifted*) search problems of the form $\widetilde{S} \coloneqq S \circ g^n$ where $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is a carefully chosen two-party gadget; see Figure 2.4. Here, Akbar and Birbal are given inputs $x \in \mathcal{X}^n$ and $y \in \mathcal{Y}^n$ respectively and their goal is find any valid solution $o \in \widetilde{S}(x,y)$, where the composed relation $\widetilde{S}$ is defined as,

$$\widetilde{S}(x,y) \;\coloneqq\; S\Big(g(x_1,y_1),\dots,g(x_n,y_n)\Big)\,.$$

Intuitively, the difficulty in solving $S \circ g^n$ arises from the fact that for any i, the i-th input bit $z_i \coloneqq g(x_i,y_i)$ to S is unknown to either party until they communicate *sufficient information* about x_i and y_i. This intuition is wrong in general for arbitrary gadgets — a simple counter-example is S being the n-bit parity function and g being the two-bit parity gadget. The goal of a *simulation theorem* is to show that for a careful choice of g, there is no better way for a protocol to solve $S \circ g^n$ other than to behave like a decision tree querying input bits of f. The first such result was proved by Raz and McKenzie [RM99] for deterministic query and communication complexity. A canonical choice for the gadget g is the Indexing gadget $\textsc{Ind}_m : [m] \times \{0,1\}^m \to \{0,1\}$ where $\textsc{Ind}_m(x,y) = y_x$, since any other gadget can be embedded into the Indexing gadget.

Theorem 2.9 ([RM99])**.** *Let $m = m(n) \coloneqq n^{\Delta}$ for $\Delta \geq 20$. For any $S \subseteq \{0,1\}^n \times \mathcal{O}$,*

$$\mathsf{FP}^{\mathsf{cc}}(S \circ \textsc{Ind}_m^n) \;=\; \Theta(\mathsf{FP}^{\mathsf{dt}}(S) \cdot \log n).$$

This paradigm has since been very successful, and lifting theorems have been proven for a variety of query and communication measures, yielding applications in diverse fields such as graph theory, learning theory (dimension complexity), combinatorial optimization (LP and SDP extended formulations), monotone computational complexity and proof complexity — we'll see applications in the last two categories in this thesis. See Table 2.1 for an overview of these lifting theorems.

M	Query	Communication	
P	Deterministic	Deterministic	[RM99, GPW15, dRNV16] [WYY17, CKLM17]
BPP	Randomized	Randomized	[GPW17]
NP	Nondeterministic	Nondeterministic	[GLM+16, Göö15]
many	Polynomial degree	Rank	[SZ09, She11, RS10]
many	Conical junta degree	Nonnegative rank	[GLM+16, KMR17]
P^{NP}	Decision lists	Rectangle overlays	[GKPW19]
PLS	Decision dags	Rectangle dags	Chapter 5
PPA	Nullstellensatz	Algebraic tiling	[PR18]
	Sherali–Adams	LP extension complexity	[CLRS13, KMR17]
	Sum-of-squares	SDP extension complexity	[LRS15]

Table 2.1: Notable query-to-communication lifting theorems.

2.6 Reductions in Query and Communication

In order to translate lower bounds on query (or communication) complexity of one search problem to lower bounds for another, we will use the following notion of reductions.

Query Reduction. A search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ *k-reduces* to $S' \subseteq \{0,1\}^m \times \mathcal{O}'$ if there exists maps $f : \{0,1\}^n \to \{0,1\}^m$ and $\phi : \mathcal{O}' \to \mathcal{O}$ such that, each bit of $f(z)$ can be computed by a depth-k decision tree on z and $o' \in S'(z) \implies \phi(o') \in S(z)$. Given any type of decision tree for S', we can run it on input $f(z)$, find a solution $o \in S'(z)$ and thereby obtain $\phi(o) \in S(z)$. Thus, $\mathsf{M}^{\mathsf{dt}}(S) \leq k \cdot \mathsf{M}^{\mathsf{dt}}(S')$ for any query complexity measure M^{dt}. In this thesis, we only consider reductions with $k = 1$, so we simply use the term *reduces* instead of 1-*reduces*.

Communication Reduction. A search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ *reduces* to $S' \subseteq \mathcal{X}' \times \mathcal{Y}' \times \mathcal{O}'$ if there exists maps $f : \mathcal{X} \to \mathcal{X}'$, $g : \mathcal{Y} \to \mathcal{Y}'$ and $\phi : \mathcal{O}' \to \mathcal{O}$ such that, $o \in S'(f(x), g(y)) \implies \phi(o) \in S(x, y)$. Given any type of protocol for S', Akbar and Birbal can locally map their inputs to $(f(x), g(y))$, run the protocol for S' to identify $o' \in S'(f(x), g(y))$ and thereby get $\phi(o') \in S(x, y)$. Thus, $\mathsf{M}^{\mathsf{cc}}(S) \leq \mathsf{M}^{\mathsf{cc}}(S')$ for any communication complexity measure M^{cc}.

Chapter 3

Tools for Lifting Theorems

In this chapter, we define some basic notations that we will use throughout this thesis. In addition, we also recall the key technical notions from prior works [GLM+16, GPW17] that are relevant for proving our lifting theorems.

Basic Notations. We always write random variables in bold (e.g. $\boldsymbol{x}, \boldsymbol{y}$). We will use the gadget $g \coloneqq \textsc{Ind}_m : [m] \times \{0,1\}^m \to \{0,1\}$ and $G \coloneqq g^n : [m]^n \times \{0,1\}^{mn} \to \{0,1\}^n$ denotes n copies of g. Capital letters X, Y usually denote sets (e.g. subsets of inputs to G) and boldface $\boldsymbol{X}$ denotes a random variable uniformly distributed over X. We will always assume that $m \geq n^{\Delta}$ for $\Delta \geq 20$. (Δ could be taken to be even smaller, but our focus is not on getting the tightest parameters.)

3.1 Rectangles are non-negative juntas

We explain how large rectangles in G's domain are related with large subcubes in G's codomain, as was first done in the foundational work of [GLM+16] (for $g : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ being the inner product gadget on $b = O(\log n)$ bits). However, as mentioned before, in our applications we will work with the Indexing gadget, the corresponding tools for which were introduced by [GPW17]. (Proving our theorems with the inner product gadget on $O(\log n)$ bits remains an interesting open problem!)

For a partial assignment $\rho \in \{0,1,*\}^n$ we let $\mathsf{free}\,\rho := \rho^{-1}(*)$ denote its *free* coordinates, and $\mathsf{fix}\,\rho := [n] \smallsetminus \mathsf{free}\,\rho$ denote its *fixed* coordinates. The number of fixed coordinates $|\mathsf{fix}\,\rho|$ is the *width* of ρ. Width-d partial assignments are naturally in 1-to-1 correspondence with width-d conjunctions: for any ρ we define $C_\rho \colon \{0,1\}^n \to \{0,1\}$ as the width-$|\mathsf{fix}\,\rho|$ conjunction that accepts an $x \in \{0,1\}^n$ iff x is consistent with ρ. Thus $C_\rho^{-1}(1) = \{x \in \{0,1\}^n : x_i = \rho_i \text{ for all } i \in \mathsf{fix}\,\rho\}$ is a subcube.

Definition 3.1. Any set $T \subseteq [m]^n \times \{0,1\}^{mn}$ is *ρ-like* if the image of T under G is precisely the subcube of n-bit strings consistent with ρ. In short,

$$T \text{ is } \rho\text{-}like \quad \iff \quad G(T) \ = \ C_\rho^{-1}(1).$$

For a random variable $\boldsymbol{x}$ we let $\mathbf{H}_\infty(\boldsymbol{x}) := \min_x \log(1/\Pr[\boldsymbol{x} = x])$ denote the usual *min-entropy* of $\boldsymbol{x}$. When $\boldsymbol{x} \in [m]^J$ for some index set J, we write $\boldsymbol{x}_I \in [m]^I$ for the marginal distribution of $\boldsymbol{x}$ on a subset $I \subseteq J$ of coordinates.

Definition 3.2 ([GLM$^+$16]). A random variable $\boldsymbol{x} \in [m]^J$ is *δ-blockwise-dense* (*δ-dense* for short) if for every non-empty $I \subseteq J$, $\boldsymbol{x}_I$ has *min-entropy rate* $\geq \delta$, that is, $\mathbf{H}_\infty(\boldsymbol{x}) \geq \delta \cdot |I| \log m$.

Definition 3.3 ([GLM$^+$16, GPW17]). Rectangle $R := X \times Y \subseteq [m]^n \times \{0,1\}^{mn}$ is *ρ-structured* if

1. $\boldsymbol{X}_{\mathsf{fix}\,\rho}$ is fixed : $\exists \{x_i^* : i \in \mathsf{fix}\,\rho\}$ such that, $x_i = x_i^*$ for each $x \in X$ and $i \in \mathsf{fix}\,\rho$.
2. $\boldsymbol{X}_{\mathsf{free}\,\rho}$ is 0.9-*dense.*
3. every $z \in G(R)$ is consistent with ρ : $G(R) \subseteq C_\rho^{-1}(1)$, that is, $g(x_i, y_i) = \rho_i$ for each $i \in \mathsf{fix}\,\rho$.
4. Y is large enough: $\mathbf{H}_\infty(\boldsymbol{Y}) \geq mn - n^3$.

An intuitive interpretation that is useful to keep in mind is that if a rectangle R is ρ-structured then it means that in narrowing down from $[m]^n \times \{0,1\}^{mn}$ to R, Akbar and Birbal have "queried" $z := G(x,y)$ all the coordinates in $\mathsf{fix}\,\rho$ and found it to be consistent with ρ, but have communicated "very little information" about the coordinates in $\mathsf{free}\,\rho$ so that they have no clue about the value of $z_{\mathsf{free}\,\rho}$ (for $z \in G(R)$). The following lemma formalizes this intuition.

Lemma 3.4 ([GKPW19, GPW17]). *For $m \geq n^\Delta$, every ρ-structured rectangle is ρ-like.*

We will however need a slight strengthening of Lemma 3.4, that for a ρ-structured R, in fact there is a *single row* of R that is already ρ-like.

Lemma 3.5 (Full Support Lemma)**.** *Let $X \times Y$ be ρ-structured. For $m \geq n^{\Delta}$, there exists $x \in X$ such that $\{x\} \times Y$ is ρ-like.*

We prove this lemma in the next section. The details of the proof are not relevant for the rest of the thesis, so the reader could feel free to skip the rest of this chapter.

We remark that the *only* reason why our lifting theorems require a gadget size of $m \geq n^{\Delta}$ is because of Lemma 3.5.

3.2 Proof of Full Support Lemma

We prove Lemma 3.5, which is a strengthening of Lemma 3.4 that was first proved in [GPW17]. For sake of completeness, we provide a direct self-contained proof of Lemma 3.5.

We first recall a simple claim from [GPW17] (also providing a proof for completeness). In what follows, let $\chi(z) \coloneqq (-1)^{\sum_i z_i}$ be the Parity function.

Claim 3.6 (cf. [GPW17, Lemma 9])**.** *If a random variable $\boldsymbol{z}$ over $\{0,1\}^n$ satisfies $|\mathbb{E}[\chi(\boldsymbol{z}_I)]| \leq n^{-2|I|}$ for every nonempty $I \subseteq J$ (for any $J \subseteq [n]$), then $\boldsymbol{z}_J$ has full support over $\{0,1\}^J$.*

Proof. For ease of notation let $J = [n]$. We can write $D(z) \coloneqq \Pr[\boldsymbol{z} = z]$ in the Fourier basis as,

$$D(z) \;=\; \sum_{I \subseteq [n]} \widehat{D}(I)\chi(z_I),$$

where $\widehat{D}(I) \coloneqq 2^{-n} \cdot \sum_z D(z)\chi(z_I) = 2^{-n} \cdot \mathbb{E}_{\boldsymbol{z} \sim D}\, \chi(\boldsymbol{z}_I)$ is the Fourier coefficient corresponding to subset I. The condition on random variable $\boldsymbol{z}$ implies that for each I, $|\widehat{D}(I)| \leq 2^{-n} \cdot n^{-2|I|}$. Thus, for any $z \in \{0,1\}^n$, we have that,

$$|D(z) - 2^{-n}| \;\leq\; \sum_{I \subseteq [n]} |\widehat{D}(I)| \;\leq\; \sum_{k=1}^{n} \binom{n}{k} \cdot n^{-2k} \cdot 2^{-n} \;\leq\; 2^{-n} \cdot \sum_{k=1}^{n} n^{-k} \;\leq\; \frac{2}{n} \cdot 2^{-n}$$

Thus, $D(z) \geq \left(1 - \frac{2}{n}\right) \cdot 2^{-n} > 0$ for each z and hence $\boldsymbol{z}$ has full support. □

To prove Lemma 3.5 we need to show that for any ρ-structured $X \times Y$, there exists $x \in X$ such that $G(\{x\} \times Y) = C_\rho^{-1}(1)$. By Claim 3.6, it suffices to show that there exists an $x \in X$ such that,

$$\forall I \subseteq \mathsf{free}\,\rho,\ I \neq \emptyset : \qquad \left| \mathbb{E}_{\mathbf{Y}} \left[\chi(g^I(x_I, \mathbf{Y}_I)) \right] \right| \leq n^{-2|I|}$$

Let's call an x satisfying the above as *good*. In order to prove that there exists a *good* x, we use a claim from [GPW17] (which was used to prove Lemma 3.4), but in a slightly strengthened form.

Claim 3.7 (Strengthening [GPW17, Lemma 8])**.** *For any ρ-structured $X \times Y$ with $\mathsf{free}\,\rho =: J \subseteq [n]$,*

$$\forall I \subseteq J,\, I \neq \emptyset : \qquad \mathop{\mathbb{E}}_{\boldsymbol{X}} \left| \mathop{\mathbb{E}}_{\mathbf{Y}} \left[\chi(g^I(\boldsymbol{X}_I, \mathbf{Y}_I)) \right] \right| \ \leq \ n^{-4|I|}.$$

Before proving Claim 3.7, we first show how this enables us to prove Lemma 3.5. By applying Markov's inequality to Claim 3.7, we have for a uniform random $\boldsymbol{x} \sim X$ and any $\emptyset \neq I \subseteq J$ that

$$\Pr_{\boldsymbol{x} \sim X} \left[\left| \mathbb{E}_{\mathbf{Y}}[\chi(g^I(\boldsymbol{x}_I, \mathbf{Y}_I))] \right| > n^{-2|I|} \right] \ \leq \ n^{-2|I|}.$$

Taking a union bound over all non-empty $I \subseteq J$, we get

$$\begin{aligned} \Pr_{\boldsymbol{x} \sim X} [\, \boldsymbol{x} \text{ is not } \textit{good} \,] \ &\leq \ \sum_{\emptyset \neq I \subseteq J} \Pr_{\boldsymbol{x} \sim X} \left[\left| \mathbb{E}_{\mathbf{Y}}[\chi(g^I(\boldsymbol{x}_I, \mathbf{Y}_I))] \right| > n^{-2|I|} \right] \\ &\leq \ \sum_{\emptyset \neq I \subseteq J} n^{-2|I|} \ = \ \sum_{d=1}^{|J|} \binom{|J|}{d} \cdot n^{-2d} \\ &\leq \ \sum_{d=1}^{|J|} n^{-d} \ \leq \ \frac{2}{n}. \end{aligned}$$

Hence most $x \in X$ are *good*. This completes the proof of Lemma 3.5, except the proof of Claim 3.7, which we now prove. This strengthened version is in fact already implicit in the original proof of Lemma 3.4 [GPW17]. For completeness, we describe a short and elegant proof given by Xinyu Wu [Wu17] and also independently by James Lee, Raghu Meka and Thomas Vidick [LMV17]. The author is grateful to Xinyu and Raghu for allowing him to include their proof in this thesis.

We prove a more tunable claim from which we can recover Claim 3.7 by setting parameters.

Claim 3.8 (Tunable version of Claim 3.7)**.** *For any rectangle $X \times Y \subseteq [m]^k \times \{0,1\}^{mk}$ such that $\mathbf{H}_\infty(\boldsymbol{X}) \geq \delta k \log m$ and $\mathbf{H}_\infty(\boldsymbol{Y}) \geq mk - s$*

$$\mathop{\mathbb{E}}_{\boldsymbol{X}} \left| \mathop{\mathbb{E}}_{\boldsymbol{Y}} \left[\chi(g^k(\boldsymbol{X}, \boldsymbol{Y})) \right] \right| \;\leq\; 4 \left(\frac{s}{m^{\delta/2}} \right)^k$$

Proof. Let $P(x) := \Pr[\boldsymbol{X} = x]$, $Q(y) := \Pr[\boldsymbol{Y} = y]$ and $N := mk$. We will interpret $y \in Y$ as a vector in $\{0,1\}^N$ indexed by tuples in $[k] \times [m]$. Define $F : \{0,1\}^N \to \mathbb{R}$ as $F(y) := 2^N \cdot Q(y)$. Thus, $\mathbb{E}_{\boldsymbol{y} \sim \mathcal{U}_N}[F(\boldsymbol{y})] = 1$ where $\mathcal{U}_N$ is the uniform distribution on $\{0,1\}^N$, and $F(y) \leq 2^s$ for all y.

For any $x \in X$, we have $\mathbb{E}_{\boldsymbol{Y}}\, G(x, \boldsymbol{Y}) = \mathbb{E}[\chi(\boldsymbol{Y}_{S_x})]$, where $S_x = \{(i, x_i) : i \in [k]\}$. Observe that for any $S \subseteq [N]$, we have the Fourier coefficient $\widehat{F}(S) = \mathbb{E}[\chi(\boldsymbol{Y}_S)]$. Thus,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{X}} \left| \mathop{\mathbb{E}}_{\boldsymbol{Y}} \chi(G(\boldsymbol{X}, \boldsymbol{Y})) \right| &= \sum_{x \in X} P(x) \left| \widehat{F}(S_x) \right| \\
&\leq \left(\sum_{x \in X} P(x)^2 \right)^{0.5} \cdot \left(\sum_{x \in X} \widehat{F}(S_x)^2 \right)^{0.5} \\
&\leq m^{-\delta k/2} \cdot \left(\sum_{\substack{S \subseteq [N] \\ |S| = n}} \widehat{F}(S)^2 \right)^{0.5} \qquad (3.1)
\end{aligned}$$

where, in the last inequality, we use that $\mathbf{H}_\infty(\boldsymbol{X}) \geq \delta k \log m$ and we sum over all Fourier coefficients corresponding to subsets of size n instead of summing over only subsets S which correspond to some x.

We use tools from Fourier analysis, namely the Bonami-Beckner noise operator and the Hypercontractivity lemma. See [O'D14] for a detailed exposition on these tools. Let T_ρ be the Bonami-Beckner noise operator, given as, $T_\rho f(y) = \mathbb{E}_{\boldsymbol{y}' \sim N_\rho(y)}[f(\boldsymbol{y}')]$ where $N_\rho(y)$ is the distribution over $\{0,1\}^N$ such that all coordinates $\boldsymbol{y}'_i$ are independent and satisfy $Pr[\boldsymbol{y}'_i = y_i] = (1+\rho)/2$. The p-norm of $f : \{0,1\}^N \to \{0,1\}$ is defined as $\|f\|_p = (\mathbb{E}_{\boldsymbol{y} \sim \mathcal{U}_N} f(\boldsymbol{y})^p)^{1/p}$. We use the following two properties of the T_ρ operator:

(1) Parseval's identity. $T_\rho f(y) = \sum_{S \subseteq [N]} \rho^{|S|} \widehat{f}(S) \chi(y_S)$. Hence $\|T_\rho f\|_2^2 \;=\; \sum_{S \subseteq [N]} \rho^{2|S|} \widehat{f}(S)^2$.

(2) Hypercontractivity Lemma. $\|T_\rho f\|_2 \leq \|f\|_p$ for $\rho = \sqrt{p-1}$.

From (1), we have for any $\rho = \sqrt{p-1}$

$$\sum_{\substack{S \subseteq [N] \\ |S|=k}} \widehat{F}(S)^2 \;\le\; \rho^{-2k} \cdot \|T_\rho F\|_2^2 \tag{3.2}$$

From (2), we have that,

$$\|T_\rho F\|_2^2 \;\le\; \|F\|_p^2 \;=\; \left(\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{U}_N} F(\boldsymbol{y})^p\right)^{2/p} \;\le\; \|F\|_\infty^{2(p-1)/p} \cdot \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{U}_N} F(\boldsymbol{y}) \;\le\; 2^{2s(p-1)/p} \tag{3.3}$$

where the last inequality uses that $\|F\|_\infty \le 2^s$ and that $\mathbb{E}_{\boldsymbol{y} \sim \mathcal{U}_N} F(\boldsymbol{y}) = 1$.

Putting together Equation 3.1, 3.2 and 3.3, and setting $p = s/(s-1)$, we get,

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{X}} \left| \mathop{\mathbb{E}}_{\boldsymbol{Y}} G(\boldsymbol{X}, \boldsymbol{Y}) \right| \;&\le\; m^{-\delta k/2} \cdot (p-1)^{-k} \cdot 2^{2s(p-1)/p} \\ &<\; 4 \left(\frac{s}{m^{\delta/2}}\right)^k . \end{aligned}$$

□

Proof of Claim 3.7. For each non-empty $I \subseteq J \coloneqq \mathsf{free}\,\rho$, we invoke Claim 3.8 with parameters $k = |I|$, $s = n^3$ and $\delta = 0.9$ applied to the marginal distributions $(\boldsymbol{X}_I, \boldsymbol{Y}_I)$. It follows from 0.9-*dense*-ness of $\boldsymbol{X}$ that $\mathbf{H}_\infty(\boldsymbol{X}_I) \ge 0.9|I| \log m$. We also have that $\mathbf{H}_\infty(\boldsymbol{Y}_I) \ge \mathbf{H}_\infty(\boldsymbol{Y}) - m(n - |I|) \ge m|I| - n^3$. Thus all requirements of Claim 3.8 are met. By taking $m \ge n^{20}$ we indeed have the claim as desired, since $\frac{s}{m^{\delta/2}} < \frac{n^{-4}}{4}$. □

Chapter 4

Dag-like Models and Equivalences

In this chapter, we introduce dag-like models in query and communication complexity and describe their connections to proof complexity and monotone computational complexity respectively.

We define all query and communication models as solving *search problems*, defined by a relation $S \subseteq \mathcal{I} \times \mathcal{O}$, for some finite input and output sets $\mathcal{I}$ and $\mathcal{O}$. On input $z \in \mathcal{I}$ the search problem is to find some output in $S(z) \coloneqq \{o \in \mathcal{O} : (z, o) \in S\}$. For convenience, we also define $S^{-1}(o) \coloneqq \{z \in \mathcal{I} : (z, o) \in S\}$.

In this thesis, we always assume that S is *total* so that $S(z) \neq \emptyset$ for all $z \in \mathcal{I}$. In particular, we will consider CSP search problems (Definition 2.5) in the query model and monotone Karchmer–Wigderson search problems (Definition 2.6) in the communication model.

4.1 Abstract dags

To motivate our definition of a dag-model, we first abstractly formalize "line-based" proof systems, parameterized by a family of functions $\mathcal{G}$ to which each line of the proof belongs. For example, as we will see later in more detail, in the case of Resolution refutations, we have $\mathcal{G} = \{\text{clauses over literals } z_1, \ldots, z_n\}$ and in the case of Cutting Planes refutations, we have $\mathcal{G} = \{\text{linear threshold functions over } z_1, \ldots, z_n\}$.

Definition 4.1 (Bottom-up definition)**.** Let $\mathcal{G}$ be a family of functions $\{0,1\}^n \to \{0,1\}$. A (semantic) *$\mathcal{G}$-refutation* of an n-variable CNF contradiction $F = \bigwedge_{i\in[m]} D_i$ is a directed acyclic graph of fan-out ≤ 2 where each node (or *line*) v is associated with a function $g_v \in \mathcal{G}$ satisfying the following:

1. *Root:* There is a distinguished root node r (fan-in 0), and $g_r \equiv 0$ is the constant 0 function.
2. *Non-leaves:* For each non-leaf node v with children u,w, we have $g_v^{-1}(1) \supseteq g_u^{-1}(1) \cap g_w^{-1}(1)$.
3. *Leaves:* Each leaf node v is labeled with a clause D of F such that $g_v^{-1}(1) \supseteq D^{-1}(1)$.

The *length* (or size) of a $\mathcal{G}$-refutation is its number of nodes. We will instead work with a *top-down* definition of dag-like models where the goal is solve total search problems. This definition has two benefits: (i) we get an "algorithmic interpretation" of a proof of refutation and (ii) generalizes to arbitrary total search problems in both query and communication models.

A version of the following definition (with a specialized $\mathcal{F}$) was introduced by [Raz95] and subsequently simplified in [Pud10, Sok17].

Definition 4.2 (Top-down definition)**.** Let $\mathcal{F}$ be a family of functions $\mathcal{I} \to \{0,1\}$. An *$\mathcal{F}$-dag* solving $S \subseteq \mathcal{I} \times \mathcal{O}$ is a directed acyclic graph of fan-out ≤ 2 where each node v is associated with a function $f_v \in \mathcal{F}$ (we call $f_v^{-1}(1)$ the *feasible set* for v) satisfying the following:

1. *Root:* There is a distinguished root node r (fan-in 0), and $f_r \equiv 1$ is the constant 1 function.
2. *Non-leaves:* For each non-leaf node v with children u,w, we have $f_v^{-1}(1) \subseteq f_u^{-1}(1) \cup f_w^{-1}(1)$.
3. *Leaves:* Each leaf node v is labeled with an output $o_v \in \mathcal{O}$ such that $f_v^{-1}(1) \subseteq S^{-1}(o_v)$.

If we specialize S to be a CSP Search Problem $S(F)$ for an unsatisfiable CNF $F = \bigwedge_i D_i$, the above specializes to the bottom-up definition of $\mathcal{G}$-refutations for $\mathcal{G} := \{\neg f : f \in \mathcal{F}\}$, that is, a proof system whose lines belong to the family of *negations* of functions in $\mathcal{F}$.

The *size* of an $\mathcal{F}$-dag is its number of nodes. For convenience, we will often define top-down dags by associating each node v with a feasible set R_v and it should be understood that $f_v(z) := \mathbf{1}\{z \in R_v\}$ is the indicator function of R_v.

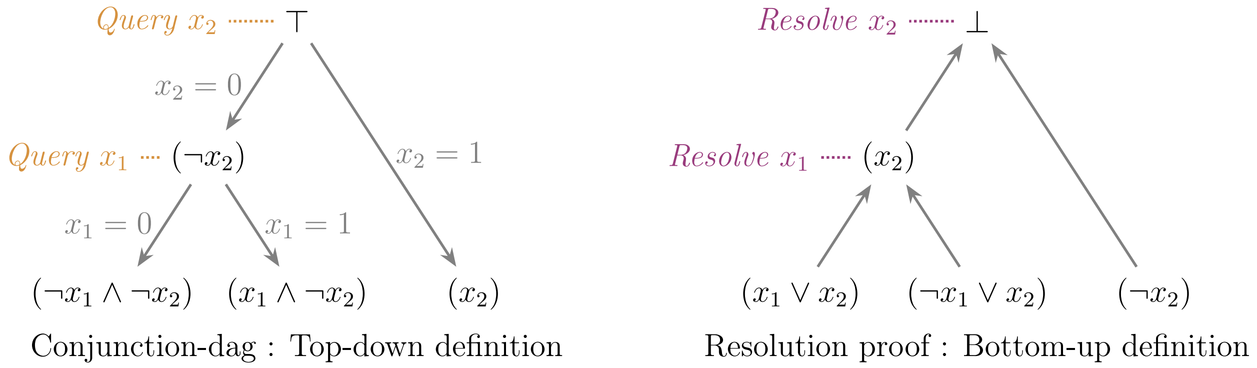


Figure 4.1: Two equivalent ways to view a Resolution refutation, illustrated in the tree-like case (see [Juk12, §18.2] for more discussion of the tree-like case).

4.2 Concrete dags

We now instantiate the abstract model for the purposes of query and communication complexity. In query complexity, the input domain of the search problem is $\mathcal{I} = \{0,1\}^n$, whereas in communication complexity, there is a fixed factorization of the input domain as $\mathcal{I} = \mathcal{X} \times \mathcal{Y}$.

4.2.1 Query dags

Conjunction-dags (essentially Resolution). Consider the n-bit input domain $\mathcal{I} \coloneqq \{0,1\}^n$ and let $\mathcal{F}$ be the set of all *conjunctions* of literals over the n input variables. Call such $\mathcal{F}$-dags simply *conjunction-dags*. We define the *width* of a conjunction-dag Π as the maximum width of a conjunction associated with a node of Π. For a search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ we define

$$\begin{aligned} \mathsf{conj\text{-}dag}(S) &\coloneqq \text{least } \textit{size} \text{ of a conjunction-dag that solves } S, \\ w(S) &\coloneqq \text{least } \textit{width} \text{ of a conjunction-dag that solves } S. \end{aligned}$$

It is evident from the equivalence between bottom-up and top-down definitions of dags in the context of CNF search problems $S = S(F)$, that conjunction-dags are equivalent to Resolution refutations; see also Figure 4.1. Indeed, $\mathsf{conj\text{-}dag}(S(F))$ is just the Resolution refutation length complexity of F, and $w(S(F))$ is the Resolution width complexity of F (cf. [BW01]).

Threshold-dags (essentially Cutting Planes). Consider the n-bit input domain $\mathcal{I} \coloneqq \{0,1\}^n$ and let $\mathcal{F}$ be the set of all *linear threshold functions* of literals over the n input variables, that is, each $f \in \mathcal{F}$ is defined by some $(n+1)$-tuple $a \in \mathbb{R}^{n+1}$ so that $f(x) = 1$ iff $\sum_{i\in[n]} a_i x_i > a_{n+1}$. Call such $\mathcal{F}$-dags simply *threshold-dags*. For a search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ we define

$$\mathsf{thresh\text{-}dag}(S) \;\coloneqq\; \text{least } \textit{size} \text{ of a threshold-dag that solves } S.$$

Similar to the case of Resolution, in the context of CNF search problems $S = S(F)$, threshold-dags are equivalent to (semantic) Cutting Planes refutations. That is, $\mathsf{thresh\text{-}dag}(S(F))$ is just the (semantic) Cutting Planes refutation length complexity of F.

Lemma 4.3 (Query dags and Proof systems)**.** *For all unsatisfiable CNFs F,*

1. $w(S(F)) \;=\; \text{Resolution-width}(F)$
2. $\mathsf{conj\text{-}dag}(S(F)) \;=\; \text{Resolution-length}(F)$
3. $\mathsf{thresh\text{-}dag}(S(F)) \;=\; \text{Cutting-Planes-length}(F)$

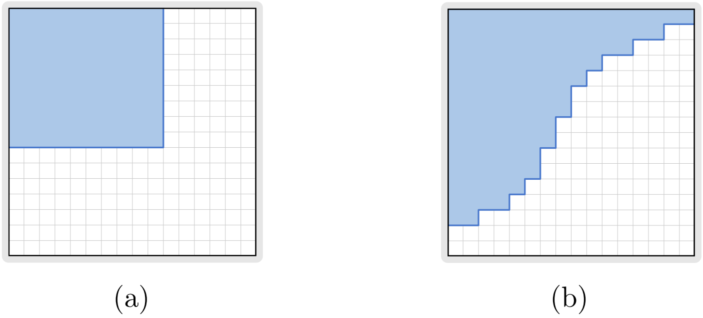
4.2.2 Communication dags

Rectangle-dags (dag-like protocols). Consider a bipartite input domain $\mathcal{I} \coloneqq \mathcal{X} \times \mathcal{Y}$ so that Akbar holds $x \in \mathcal{X}$, Birbal holds $y \in \mathcal{Y}$, and let $\mathcal{F}$ be the set of all indicator functions of *(combinatorial) rectangles over $\mathcal{X} \times \mathcal{Y}$* (sets of the form $X \times Y$ with $X \subseteq \mathcal{X}$, $Y \subseteq \mathcal{Y}$). Call such $\mathcal{F}$-dags simply *rectangle-dags*. For a search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ we define its *rectangle-dag complexity* by

$$\mathsf{rect\text{-}dag}(S) \;\coloneqq\; \text{least } \textit{size} \text{ of a rectangle-dag that solves } S.$$

In circuit complexity, a straightforward generalization of the Karchmer–Wigderson depth characterization [KW88] shows that the monotone circuit complexity of any monotone function f equals $\mathsf{rect\text{-}dag}(\mathsf{mKW}(f))$ (cf. [Pud10, Sok17]). For completeness, we provide a proof in Lemma 4.4.

In proof complexity, a useful-to-study semantic proof system is captured by *$\mathcal{F}_c$-dags solving CNF search problems $S(F)$* where $\mathcal{F}_c$ is the family of all functions $\mathcal{X}\times\mathcal{Y} \to \{0,1\}$ (where $\mathcal{X}\times\mathcal{Y} = \{0,1\}^n$ corresponds to a bipartition of the n input variables of $S(F)$) that can be computed by tree-like


(a)

(b)

Figure 4.2: We study communication dags whose feasible sets are (a) *rectangles* or (b) *triangles*. In (b), the rows are sorted in decreasing order of a_T while columns, in increasing order of b_T.

protocols of communication cost c, say for $c = \text{polylog}(n)$. Such a proof system can simulate other systems (such as Resolution and Cutting Planes with bounded coefficients), and hence lower bounds against $\mathcal{F}_c$-dags imply lower bounds against other concrete proof systems. Moreover, any $\mathcal{F}_c$-dag can be simulated by a rectangle-dag with at most a factor 2^c blow-up in size, and hence we do not lose in generality by studying only rectangle-dags.

Triangle-dags. For a bipartite input domain $\mathcal{I} := \mathcal{X} \times \mathcal{Y}$ and let $\mathcal{F}$ be the set of all indicator functions of *(combinatorial) triangles over* $\mathcal{X} \times \mathcal{Y}$; here a *triangle* $T \subseteq \mathcal{X} \times \mathcal{Y}$ is a set that can be written as $T = \{(x, y) \in \mathcal{X} \times \mathcal{Y} : a_T(x) > b_T(y)\}$ for some labeling of the rows $a_T \colon \mathcal{X} \to \mathbb{R}$ and columns $b_T \colon \mathcal{Y} \to \mathbb{R}$ by real numbers; see Figure 4.2b. In particular, every rectangle is a triangle. Call such $\mathcal{F}$-dags simply *triangle-dags*. For a search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ we define

$$\mathsf{tri\text{-}dag}(S) \;\; := \;\; \text{least } \textit{size} \text{ of a triangle-dag that solves } S.$$

Hrubeš and Pudlák [HP18] showed recently that the *monotone real circuit complexity* of an f equals $\mathsf{tri\text{-}dag}(\mathsf{mKW}(f))$; for completeness we provide a proof in Lemma 4.4.

Lemma 4.4 (Communication dags and Circuits)**.** *For all monotone* $f : \{0, 1\}^n \to \{0, 1\}$*,*

1. $\mathsf{rect\text{-}dag}(\mathsf{mKW}(f)) \;=\; \textit{monotone circuit size}(f)$*, and*

   $\mathsf{rect\text{-}dag}(\mathsf{KW}(f)) \;=\; \Theta(\textit{circuit size}(f))$*.*
2. $\mathsf{tri\text{-}dag}(\mathsf{mKW}(f)) \;=\; \textit{monotone real-circuit size}(f)$*.*

The complexity measures introduced so far are related as follows; here S' is *any* two-party version of S obtained by choosing some factorization $\mathcal{X} \times \mathcal{Y} = \{0,1\}^n$ of the input domain of S:

$$\mathsf{tri\text{-}dag}(S') \quad \leq \quad \left\{ \begin{matrix} \mathsf{rect\text{-}dag}(S') \\ \mathsf{thresh\text{-}dag}(S) \end{matrix} \right\} \quad \leq \quad \mathsf{conj\text{-}dag}(S) \quad \leq \quad n^{O(w(S))}. \tag{4.1}$$

The first two inequalities holds because each conjunction can be simulated by a rectangle as well as by a linear threshold function, either of which can be simulated by a triangle. The last inequality holds since there are at most $n^{O(w)}$ many distinct width-w conjunctions, and we may assume w.l.o.g. that any $f \in \mathcal{F}$ is associated with at most one node in an $\mathcal{F}$-dag (any incoming edge to a node v can be rewired to the *lowest* node u, in topological order, such that $f_v = f_u$).

4.3 Equivalence of Circuits and Communication Dags

We prove Lemma 4.4 which characterizes monotone complexity measures in terms of communication dags. These characterizations are already explicit in literature. However, since our definitions are stated in a slightly different (but equivalent) way as compared to prior literature, we provide proofs of these characterizations for completeness.

For convenience, we will define top-down dags by associating each node v with a feasible set R_v and it should be understood that $f_v(z) \coloneqq \mathbf{1}\{z \in R_v\}$ is the indicator functions of R_v.

Proof of Lemma 4.4. We begin with (1.) and, for simplicity, only deal with case of monotone circuits for now. These characterizations already exist in literature due to [Raz95, Sok17].

Circuits $\longrightarrow$ Rectangle-Dags. Given a monotone circuit C computing a function $f : \{0,1\}^n \to \{0,1\}$, we will construct a rectangle-dag solving $\mathsf{mKW}(f)$ with the same dag structure as C. For any gate g, we assign it a rectangular feasible set $R_g \coloneqq X_g \times Y_g$ where $X_g \coloneqq \{x \in f^{-1}(1) : g(x) = 1\}$ and $Y_g \coloneqq \{y \in f^{-1}(0) : g(y) = 0\}$. We label a leaf node computing x_i by the output label $i \in [n]$. We now verify the three properties of rectangle-dags:

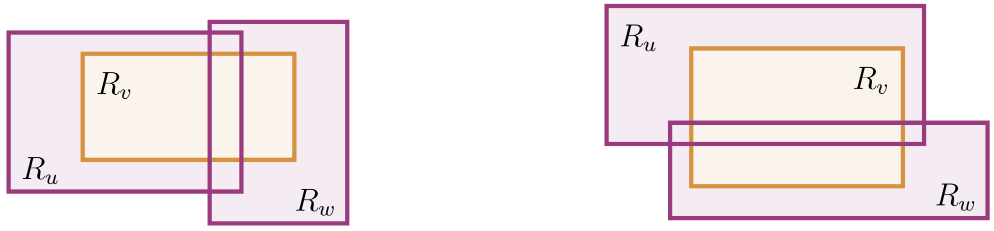
1. *Root:* Since the root gate g computes f, we have $R_g = f^{-1}(1) \times f^{-1}(0)$.

2. *Non-leaves:* Suppose gate g has children g_1, g_2. If $g = g_1 \wedge g_2$, then $X_g = X_{g_1} \cap X_{g_2}$ and $Y_g = Y_{g_1} \cup Y_{g_2}$. If $g = g_1 \vee g_2$, then $X_g = X_{g_1} \cup X_{g_2}$ and $Y_g = Y_{g_1} \cap Y_{g_2}$. In either case, we have $R_g \subseteq R_{g_1} \cup R_{g_2}$ as desired.
3. *Leaves:* Consider a leaf gate g computing the function z_i for some $i \in [n]$. By definition of R_g, we have that $x_i = 1$ and $y_i = 0$ for all $(x, y) \in R_g$, that is, $R_g \subseteq \mathsf{mKW}(f)^{-1}(i)$.

Rectangle-Dags $\longrightarrow$ Circuits. Given a rectangle-dag solving $\mathsf{mKW}(f)$, we construct a monotone circuit C with the same dag structure, that computes f. In place of leaves labeled with output $i \in [n]$, we include gates computing the input variable z_i. For every non-leaf node v with children u and w, we have associated feasible sets such that $R_v \subseteq R_u \cup R_w$. Let $R_v =: X_v \times Y_v$. The key observation is that either (a) $X_v \subseteq X_u \cap X_w$ or (b) $Y_v \subseteq Y_u \cap Y_w$ holds (cf. Figure 4.3). If (a) holds, we include an $\wedge$ gate and if (b) holds, we include an $\vee$ gate in place of v.

We prove correctness of the circuit by induction on nodes of the rectangle-dag, with the following inductive hypothesis: for any node v, the function g_v computed by the corresponding sub-circuit under v satisfies that $g_v^{-1}(1) \supseteq X_v$ and $g_v^{-1}(0) \supseteq Y_v$.

1. *Leaves (base case):* A leaf node v labeled by $i \in [n]$ is replaced by gate z_i. But we know that $x_i = 1$ for all $x \in X_v$ and $y_i = 0$ for all $y \in Y_v$.
2. *Non-Leaves (induction):* Suppose (a) $X_v \subseteq X_u \cap X_w$ holds and we have an $\wedge$ gate in place of v. By inductive hypothesis, we have that $g_u(x) = 1$ for all $x \in X_u$ and $g_w(1) = 1$ for all $x \in X_w$, hence, we have that $g_v(x) \coloneqq g_u(x) \wedge g_w(x) = 1$ for all $x \in X_v \subseteq X_u \cap X_w$. On the other hand, we have that $g_u(y) = 0$ for all $y \in Y_u$ and $g_w(y) = 0$ for all $y \in Y_w$, hence, we have that $g_v(y) \coloneqq g_u(y) \wedge g_w(y) = 0$ for all $y \in Y_v \subseteq Y_u \cup Y_w$. Case (b) is handled similar.



(a) $X_v \subseteq X_u \cap X_w$ (b) $Y_v \subseteq Y_u \cap Y_w$

Figure 4.3: Only two possible scenarios for $R_v \subseteq R_u \cup R_w$

3. *Root:* For the root gate v we have that $g_v^{-1}(1) \supseteq f^{-1}(1)$ and $g_v^{-1}(0) \supseteq f^{-1}(0)$. Thus, $g_v \equiv f$.

Case of non-monotone circuits. The procedure to convert rectangle-dags solving $\mathsf{KW}(f)$ to (non-monotone) circuits computing f also works similarly, with the only change being how we label the leaves. If a leaf node v is labeled with $i \in [n]$, then either $x_i = 1$ and $y_i = 0$ for all $(x, y) \in R_v$ or $x_i = 0$ and $y_i = 1$ for all $(x, y) \in R_v$. In the former case, we include the gate z_i, and in the latter case, the gate $\neg z_i$ in place of v. Thus, *circuit size*$(f) \leq \mathsf{rect\text{-}dag}(\mathsf{KW}(f))$.

To convert non-monotone circuits into rectangle-dags, we first pre-process the given non-monotone circuit C into a circuit C' such that $\text{size}(C') \leq 2 \cdot \text{size}(C)$ and all the NOT gates are at the leaves. We can now perform the transformation of going from circuits to rectangle-dags as described before. This shows that $\mathsf{rect\text{-}dag}(\mathsf{KW}(f)) \leq 2 \cdot (\textit{circuit size}(f))$.

We now move to (2.) the equivalence between real monotone circuits and triangle dags. This characterization was proved only recently in [HP18].

Real Monotone Circuits $\longrightarrow$ Triangle-Dags. Given a real monotone circuit C computing a monotone $f : \{0,1\}^n \to \{0,1\}$, we construct a $\mathsf{tri\text{-}dag}$ with the same dag structure as C solving $\mathsf{mKW}(f)$. For any gate g, we assign it a triangular feasible set $T_g \coloneqq \{(x, y) : g(x) > g(y)\}$. We now verify the three properties of triangle-dags:

1. *Root:* Root gate g computes f and hence $T_g = \{(x, y) : g(x) > g(y)\} = f^{-1}(1) \times f^{-1}(0)$.
2. *Non-leaves:* Suppose gate g has children g_1, g_2. That is, $g(z) = \Phi_g(g_1(z), g_2(z))$, where $\Phi_g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a monotone function. For any $(x, y) \in T_g$, since $g(x) > g(y)$ and Φ_g is monotone, at least one of $g_1(x) > g_1(y)$ or $g_2(x) > g_2(y)$ must hold. Thus, $(x, y) \in T_{g_1} \cup T_{g_2}$.
3. *Leaves:* Consider a leaf gate g computing the function z_i for some $i \in [n]$. By definition of T_g, we have that $x_i = 1$ and $y_i = 0$ for all $(x, y) \in T_g$. Thus, $T_g \subseteq \mathsf{mKW}(f)^{-1}(i)$.

Triangle-Dags $\longrightarrow$ Real Monotone Circuits. Given a $\mathsf{tri\text{-}dag}$ solving $\mathsf{mKW}(f)$, we construct a real monotone circuit C with the same dag structure, that computes f. For every node v let $a_v : f^{-1}(1) \to \mathbb{R}$ and $b_v : f^{-1}(0) \to \mathbb{R}$ be the functions that define the feasible set $T_v =$

$\{(x, y) : a_v(x) > b_v(y)\}$. For leaves labeled with output $i \in [n]$, we can assume, without loss of generality, that $a_v(x) = x_i$ and $b_v(y) = y_i$.[1]

In place of leaves labeled with output $i \in [n]$, we include gates computing the input variable z_i. For every non-leaf node, we include a monotone gate $\Phi_v : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, in a way such that,

$$\Phi_v(\alpha_1, \alpha_2) = \max_{z \in f^{-1}(1)} a_v(z) \quad \text{subject to, } a_u(z) \leq \alpha_1 \text{ and } a_w(z) \leq \alpha_2$$

We prove correctness of the circuit by induction on nodes of the triangle-dag, with the following inductive hypothesis: for any node v, the function g_v computed by the corresponding sub-circuit under v satisfies that $g_v(x) \geq a_v(x)$ and $g_v(y) \leq b_v(y)$ for all $(x, y) \in f^{-1}(1) \times f^{-1}(0)$.

1. *Leaves (base case):* All leaf nodes v labeled by $i \in [n]$ correspond to the gate $g_v(z) = z_i$, which is same as $a_v(z)$ if $z \in f^{-1}(1)$, or $b_v(z)$ if $z \in f^{-1}(0)$.
2. *Non-Leaves (induction):* It is easy to see that $g_v(x) \geq a_v(x)$ for all $x \in f^{-1}(1)$ by setting $z = x$. Suppose for contradiction that there exists a $y \in f^{-1}(0)$ such that $g_v(y) > b_v(y)$. Hence, there exists a $z \in f^{-1}(1)$ such that $a_v(z) > b_v(y)$ and hence $(z, y) \in T_v$. But on the other hand, by the inductive hypothesis we have that $a_u(z) \leq g_u(y) \leq b_u(y)$ and $a_w(z) \leq g_w(y) \leq b_w(y)$. Hence, we have that $(z, y) \notin T_u \cup T_w$, which is a contradiction.
3. *Root:* For the root gate v, we have that $g_v(x) \geq a_v(x) > b_v(y) \geq g_v(y)$ for all $(x, y) \in f^{-1}(1) \times f^{-1}(0)$. Thus, we can choose a threshold t between $\min_{x \in f^{-1}(1)} g_v(x)$ and $\max_{y \in f^{-1}(0)} g_v(y)$ and compute $f(z) = \mathbf{1}\{g_v(z) \geq t\}$. Note that this doesn't require any additional gate.

□

[1] By letting $a_v(x) = x_i$ and $b_v(y) = y_i$, we are only enlarging T_v without affecting correctness.

Chapter 5

Monotone Circuit Lower Bounds

In this chapter we prove our first lifting theorem, establishing a characterization of the rectangle-dag complexity for *composed* search problems of the form $S \circ g^n$, in terms of the conjunction-dag width of S. This show that the corresponding bound Equation 4.1 (in Chapter 4) is tight for such composed search problems.

Theorem 5.1. *Let $m = m(n) \coloneqq n^{\Delta}$ for $\Delta \geq 20$. For any $S \subseteq \{0,1\}^n \times \mathcal{O}$,*

$$\textsf{rect-dag}(S \circ \textrm{I}\small{\textrm{ND}}^n_m) \;=\; n^{\Theta(w(S))}.$$

As an implication, we get that for any CNF contradiction F that requires large Resolution width to refute, there is an *explicit* associated monotone function $f : \{0,1\}^N \to \{0,1\}$ that requires large monotone circuits to compute.

Corollary 5.2. *Any n-variate unsatisfiable k-CNF formula F with ℓ clauses that requires resolution width w to refute is associated with an explicit monotone function $f : \{0,1\}^N \to \{0,1\}$ on $N = \ell \cdot n^{\Delta k}$ variables, which requires monotone circuits of size $n^{\Omega(w)}$.*

In particular, starting with a CNF contradiction with $\ell = O(n)$, $k = O(1)$, and $w = \Omega(n)$ get that the associated monotone function on $N = n^{O(1)}$ variables requires monotone circuits of size $2^{N^{\Omega(1)}}$. An unsatisfactory thing about this corollary however is that we don't get a "nice" description of the associated hard monotone function.

To overcome this issue, we provide a different reduction methodology, thereby getting an exponential lower bound on the monotone circuit size of (a monotone variant of) 3Xor-Sat, defined as follows: An input $x \in \{0,1\}^N$ is interpreted as (the indicator vector of) a set of 3Xor constraints over n boolean variables $v_1, \dots, v_n$ (there are $N = 2n^3$ possible constraints). The function $3\textsc{Xor-Sat}_n(x) := 1$ iff the set x is *unsatisfiable*, that is, no boolean assignment to the v_i satisfies all constraints in x. This is indeed a monotone function: flipping any bit of x from 0 to 1, is tantamount to adding a new constraint to the instance, making it even harder to satisfy.

Corollary 5.3. $3\textsc{Xor-Sat}_n$ *requires monotone circuits of size* $2^{n^{\Omega(1)}}$.

This theorem stands in contrast to the fact that 3Xor-Sat is computable by linear-sized monotone $\mathbb{F}_2$-span programs; this is the first such function efficiently computable by monotone span programs, that provably requires exponential monotone circuit size. Given that there exist fast parallel (non-monotone) algorithms for linear algebra [Mul87], we also have that 3Xor-Sat is in NC^2. Thus our result improves qualitatively on the monotone vs. non-monotone separation of Tardos [Tar88] who exhibited a monotone function in P (computed by solving a semidefinite program) with exponential monotone circuit complexity. For further comparison, another famous candidate problem to witness a monotone vs. non-monotone separation is the *perfect matching* function: it is in RNC^2 [Lov79] while it is widely conjectured to have exponential monotone circuit complexity (a quasipolynomial lower bound was proved by Razborov [Raz85a]).

5.1 Rectangle Partitioning Scheme

We show that given any rectangle $R := X \times Y \subseteq [m]^n \times \{0,1\}^{mn}$, we can partition most of $X \times Y$ into ρ-structured subrectangles with $|\mathsf{fix}\,\rho|$ bounded in terms of the size of $X \times Y$. Indeed, we describe a simple 2-round partitioning scheme from [GPW17] below; see also Figure 5.1. In the 1st round of the algorithm, we partition the rows as $X = \bigsqcup_i X^i$ where each X^i will be fixed on some blocks $I_i \subseteq [n]$ and 0.95-dense[1] on the remaining blocks $[n] \smallsetminus I_i$. In the 2nd round, each $X^i \times Y$ is further partitioned along columns so as to fix the outputs of the gadgets on coordinates I_i.

[1] the proof could equivalently be done with 0.9-dense-ness that would align more with the definition of *ρ-structured*-ness. However, choosing 0.95 helps us with parameters in Chapter 6

```
Rectangle Scheme
Input: $R = X \times Y \subseteq [m]^n \times \{0,1\}^{mn}$.
Output: A partition of $R$ into subrectangles.

1st round: Iterate the following for $i = 1, 2, \ldots,$ until $X$ becomes empty:
    (i) Let $I_i \subseteq [n]$ be a maximal subset (possibly $I_i = \emptyset$) such that $\boldsymbol{X}_{I_i}$ has min-entropy rate
        $< 0.95$, and let $\alpha_i \in [m]^{I_i}$ be an outcome witnessing this: $\Pr[\boldsymbol{X}_{I_i} = \alpha_i] > m^{-0.95|I_i|}$
    (ii) Define $X^i := \{x \in X : x_{I_i} = \alpha_i\}$
    (iii) Update $X \leftarrow X \smallsetminus X^i$
2nd round: For each part $X^i$ and $\gamma \in \{0,1\}^{I_i}$, define $Y^{i,\gamma} := \{y \in Y : g^{I_i}(\alpha_i, y_{I_i}) = \gamma\}$
return $\{R^{i,\gamma} := X^i \times Y^{i,\gamma} : Y^{i,\gamma} \neq \emptyset\}$
```

All properties of Rectangle Scheme that we will subsequently need are formalized below; see also Figure 5.1. For terminology, given a subset $A' \subseteq A$ we define its *density* (inside A) as $|A'|/|A|$.

Lemma 5.4 (Rectangle Lemma)**.** *Fix any parameter $k \leq n \log n$. Given a rectangle $R \subseteq [m]^n \times \{0,1\}^{mn}$, let $R = \bigsqcup_i R^i$ be the output of Rectangle Scheme. Then there exist "error" sets $X_{\mathsf{err}} \subseteq [m]^n$ and $Y_{\mathsf{err}} \subseteq \{0,1\}^{mn}$, both of density $\leq 2^{-k}$ (inside their respective sets), such that for each i, one of the following holds:*

- ▷ ***Structured case:*** *R^i is ρ^i-structured for some ρ^i of width at most $O(k/\log n)$.*
- ▷ ***Error case:*** *R^i is covered by error rows/columns, i.e., $R^i \subseteq X_{\mathsf{err}} \times \{0,1\}^{mn} \cup [m]^n \times Y_{\mathsf{err}}$.*

*A **query alignment** property holds: for every $x \in [m]^n \smallsetminus X_{\mathsf{err}}$, there exists a subset $I_x \subseteq [n]$ with $|I_x| \leq O(k/\log n)$ such that every "structured" R^i intersecting $\{x\} \times \{0,1\}^{mn}$ has $\mathsf{fix}\,\rho^i \subseteq I_x$.*

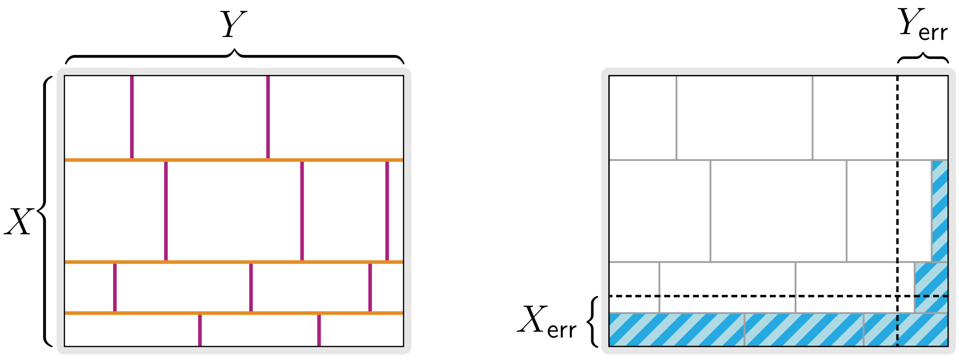
The proof is implicit in [GLM+16, GPW17]. We start by recording a basic fact about behavior of min-entropy on conditioning and a key property of the 1st round of Rectangle Scheme.

Fact 5.5. *Let $\boldsymbol{X}$ be a random variable and E an event. Then $\mathbf{H}_\infty(\boldsymbol{X} \mid E) \geq \mathbf{H}_\infty(\boldsymbol{X}) - \log 1/\Pr[E]$.*

Claim 5.6. *Each part X^i obtained in 1st round of Rectangle Scheme satisfies:*

- Blockwise-density: $\boldsymbol{X}^i_{[n]\smallsetminus I_i}$ *is* 0.95*-dense.*
- Relative size: $|X^{\geqslant i}| \leq m^{n-0.05|I_i|}$ *where* $X^{\geqslant i} := \bigcup_{j \geq i} X^j$.

Proof. By definition, $\boldsymbol{X}^i = (\boldsymbol{X}^{\geqslant i} \mid \boldsymbol{X}^{\geqslant i}_{I_i} = \alpha_i)$. Suppose for contradiction that $\boldsymbol{X}^i_{[n]\smallsetminus I_i}$ is not 0.95-dense. Then there is some nonempty subset $K \subseteq [n] \smallsetminus I_i$ and an outcome $\beta \in [m]^K$ violating the



(a)

(b)

Figure 5.1: (a) Rectangle Scheme partitions $R = X \times Y$ first along rows, then along columns. (b) Lemma 5.4 illustrated: most subrectangles are ρ-structured for low-width ρ, except some error parts (highlighted in figure) that are contained in few error rows/columns X_{err}, Y_{err}.

min-entropy condition, namely $\Pr[\boldsymbol{X}^i_K = \beta] > m^{-0.95|K|}$. But this contradicts the maximality of I_i since the larger set $I_i \cup K$ now violates the min-entropy condition for $\boldsymbol{X}^{\geqslant i}$:

$$\Pr[\boldsymbol{X}^{\geqslant i}_{I_i \cup K} = \alpha_i \beta] \;=\; \Pr[\boldsymbol{X}^{\geqslant i}_{I_i} = \alpha_i] \cdot \Pr[\boldsymbol{X}^i_K = \beta] \;>\; m^{-0.95|I_i|} \cdot m^{-0.95|K|} \;=\; m^{-0.95(|I_i \cup K|)} \,.$$

This shows the first property. For the second property, apply Fact 5.5 for $\boldsymbol{X}^i = (\boldsymbol{X}^{\geqslant i} \mid \boldsymbol{X}^{\geqslant i}_{I_i} = \alpha_i)$ to find that $\mathbf{H}_\infty(\boldsymbol{X}^i) \geq \mathbf{H}_\infty(\boldsymbol{X}^{\geqslant i}) - 0.95|I_i| \log m$. On the other hand, since $\boldsymbol{X}^i$ is fixed on I_i, we have $\mathbf{H}_\infty(\boldsymbol{X}^i) \leq (n - |I_i|) \log m$. Combining these two inequalities we get $\mathbf{H}_\infty(\boldsymbol{X}^{\geqslant i}) \leq (n - 0.05|I_i|) \log m$, which yields the second property. □

Proof of Lemma 5.4 . *Identifying* Y_{err}, X_{err}. We define $Y_{\mathsf{err}} := \bigcup_{i,\gamma} Y^{i,\gamma}$ subject to $|Y^{i,\gamma}| < 2^{mn-n^2}$. To bound the size of Y_{err}, we claim that there are at most $(4m)^n$ possible choices of i, γ. Indeed, each X^i is associated with a unique pair $(I_i \subseteq [n], \alpha_i \in [m]^{I_i})$, and there are at most 2^n choices of I_i and at most m^n choices of corresponding α_i. Also, for each X^i, there are at most 2^n possible assignments to $\gamma \in \{0,1\}^{I_i}$. For each i, γ, we add at most 2^{mn-n^2} columns to Y_{err}. Thus, Y_{err} has density at most $(4m)^n \cdot 2^{-n^2} < 2^{-k}$ inside $\{0,1\}^{mn}$.

We define $X_{\mathsf{err}} := \bigsqcup_i X^i$ subject to $|I_i| > 20k/\log m$. Let i be the least index with $|I_i| > 20k/\log m$ so that $X_{\mathsf{err}} \subseteq X^{\geqslant i}$. By Claim 5.6, $|X^{\geqslant i}| \leq m^{n-0.05|I_i|} < m^n \cdot 2^{-k}$ since $|I_i| > 20k/\log m$. In other words, $X^{\geqslant i}$, and hence X_{err}, has density at most 2^{-k} inside $[m]^n$.

Structured vs. error. Let $R^{i,\gamma} := X^i \times Y^{i,\gamma}$, where X_i is associated with (I_i, α_i), be a rectangle *not* contained in the error rows/columns. By definition of X_{err}, Y_{err}, this means $|Y^{i,\gamma}| \geq 2^{mn-n^2}$ (so that $\mathbf{H}_\infty(\boldsymbol{Y}^{i,\gamma}) \geq mn - n^2$) and $|I_i| \leq 20k/\log m$. We have from Claim 5.6 that $\boldsymbol{X}^i_{[n]\smallsetminus I_i}$ is 0.95-dense. Hence, $R^{i,\gamma}$ is ρ^i-structured where ρ^i equals γ on I_i and consists of stars otherwise.

Query alignment. For each $x \in [m]^n \smallsetminus X_{\mathsf{err}}$, we define $I_x = I_i$ where X^i is the unique part that contains x. It follows that any ρ-structured rectangle that intersects the x-th row is of the form $X^i \times Y^{i,\gamma}$ and hence has $\mathsf{fix}\,\rho = I_i$. Since $X^i \nsubseteq X_{\mathsf{err}}$, we have $|I_i| \leq O(k/\log n)$. □

5.2 Lifting for Rectangle-Dags

In this section we prove the nontrivial direction of Theorem 5.1: Let Π be a rectangle-dag solving $S \circ \textsc{Ind}^n_m$ of size n^d for some d. Our goal is to show that $w(S) \leq O(d)$.

5.2.1 Game semantics for dags

For convenience (and fun), we use the language of two-player competitive games, introduced in [Pud00, AD08], which provides an alternative way of thinking about conjunction-dags solving $S \subseteq \{0,1\}^n \times \mathcal{O}$. The game involves two competing players, *Explorer* and *Adversary*, and proceeds in rounds. The state of the game in each round is modeled as a partial assignment $\rho \in \{0,1,*\}^n$. At the start of the game, $\rho := *^n$. In each round, Explorer makes one of two moves:

- *Query a variable:* Explorer specifies an $i \in \mathsf{free}\,\rho$, and Adversary responds with a bit $b \in \{0,1\}$. The state ρ is updated by $\rho_i \leftarrow b$.
- *Forget a variable:* Explorer specifies an $i \in \mathsf{fix}\,\rho$, and the state is updated by $\rho_i \leftarrow *$.

An important detail is that Adversary is allowed to choose $b \in \{0,1\}$ freely even if the i-th variable was queried (with response different from b) and subsequently forgotten during past play. The game ends when a solution to S can be inferred from ρ, that is, when $C_\rho^{-1}(1) \subseteq S^{-1}(o)$ for some $o \in \mathcal{O}$.

Explorer's goal is to end the game while keeping the width of the game state ρ as small as possible. Indeed, Atserias and Dalmau [AD08] prove that $w(S)$ is characterized (up to an additive ±1) as the least w such that the Explorer has a strategy for ending the game that keeps the width of the game state at most w throughout the game. (A similar characterization exists for dag *size* [Pud00].) Hence our goal becomes to describe an Explorer-strategy for S such that the width of the game state never exceeds $O(d)$ regardless of how the Adversary plays.

5.2.2 Simplified proof

To explain the basic idea, we first give a simplified version of the proof: We assume that all rectangles R involved in Π—call them the *original* rectangles—can be partitioned *errorlessly* into ρ-structured subrectangles for ρ of width $O(d)$. That is, invoking Rectangle Scheme for each original R, we assume that

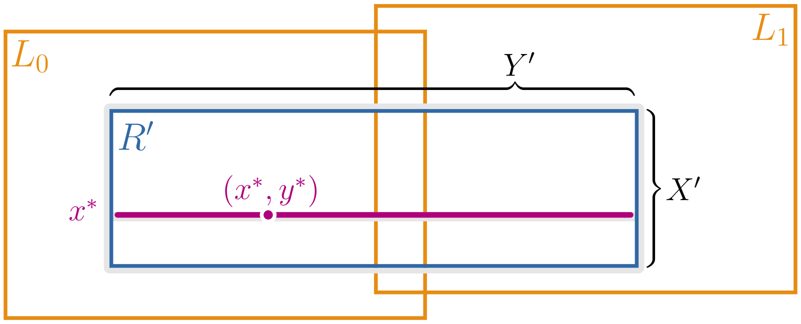
($*$) *Assumption:* All subrectangles in the partition $R = \bigsqcup_i R^i$ output by Rectangle Scheme satisfy the "structured" case of Lemma 5.4 for $k := 2d \log n$.

In Section 5.2.3 we remove this assumption by explaining how the proof can be modified to work in the presence of some error rows/columns.

Overview. We extract a width-$O(d)$ Explorer-strategy for S by walking down the rectangle-dag Π, starting at the root. For each original rectangle R that is reached in the walk, we maintain a ρ-structured subrectangle $R' \subseteq R$ chosen from the partition of R. Note that ρ will have width $O(d)$ by our choice of k. The intention is that ρ will record the current state of the game. There are three issues to address: (1) Why is the starting condition of the game met? (2) How do we take a step from a node of Π to one of its children? (3) Why are we done once we reach a leaf?

(1) Root case. At start, the root of Π is associated with the original rectangle $R = [m]^n \times \{0,1\}^{mn}$ comprising the whole domain. The partition of R computed by Rectangle Scheme is trivial: it contains a single part, the $*^n$-structured R itself. Hence we simply maintain the $*^n$-structured $R \subseteq R$, which meets the starting condition for the game.

(2) Internal step. This is the crux of the argument: Supposing the game has reached state $\rho_{R'}$ and we are maintaining some $\rho_{R'}$-structured subrectangle $R' \subseteq R$ associated with an internal node v, we want to move to some $\rho_{L'}$-structured subrectangle $L' \subseteq L$ associated with a child of v. Moreover, we must keep the width of the game state at most $O(d)$ during this move.



Since $R' =: X' \times Y'$ is $\rho_{R'}$-structured, we have from Lemma 3.5 that there exists some $x^* \in X'$ such that $\{x^*\} \times Y'$ is $\rho_{R'}$-like. Let the two original rectangles associated with the children of v be L_0 and L_1. Let $\bigsqcup_i L_b^i$ be the partition of L_b output by Rectangle Scheme. By query alignment in Lemma 5.4, there is some $I_b^* \subseteq [n]$, $|I_b^*| \leq O(d)$, such that all L_b^i that intersect the x^*-th row are ρ^i-structured with $\mathsf{fix}\,\rho^i \subseteq I_b^*$. As Explorer, we now query the input variables in coordinates $J := (I_0^* \cup I_1^*) \smallsetminus \mathsf{fix}\,\rho_{R'}$ (in any order) obtaining some response string $z_J \in \{0,1\}^J$ from the Adversary. As a result, the state of the game becomes the extension of $\rho_{R'}$ by z_J, call it ρ^*, which has width $|\mathsf{fix}\,\rho^*| = |\mathsf{fix}\,\rho_{R'} \cup J| \leq O(d)$.

Note that there is some $y^* \in Y'$ (and hence $(x^*, y^*) \in R' \subseteq L_0 \cup L_1$) such that $G(x^*, y^*)$ is consistent with ρ^*; indeed, the whole row $\{x^*\} \times Y'$ is $\rho_{R'}$-like and ρ^* extends $\rho_{R'}$. Suppose $(x^*, y^*) \in L_0$; the case of L_1 is analogous. Let $L' \subseteq L_0$ be the unique part in the partition of L_0 such that $(x^*, y^*) \in L'$. Note that L' is $\rho_{L'}$-like for some $\rho_{L'}$ that is consistent with $G(x^*, y^*)$ and $\mathsf{fix}\,\rho_{L'} \subseteq I_0^*$ (by query alignment). Hence ρ^* extends $\rho_{L'}$. As Explorer, we now forget all queried variables in ρ^* except those queried in $\rho_{L'}$.

We have recovered our invariant: the game state is $\rho_{L'}$ and we maintain a $\rho_{L'}$-structured subrectangle L' of an original rectangle L_0. Moreover, the width of the game state remained $O(d)$.

(3) Leaf case. Suppose the game state is ρ and we are maintaining an associated ρ-structured subrectangle $R' \subseteq R$ corresponding to a *leaf* node. The leaf node is labeled with some solution $o \in \mathcal{O}$ satisfying $R' \subseteq (S \circ G)^{-1}(o)$, that is, $G(R') \subseteq S^{-1}(o)$. But $G(R') = C_\rho^{-1}(1)$ by Lemma 3.4 so that $C_\rho^{-1}(1) \subseteq S^{-1}(o)$. Therefore the game ends. This concludes the (simplified) proof.

5.2.3 Accounting for error

Next, we explain how to get rid of the assumption $(*)$ by accounting for the rows and columns that are classified as error in Lemma 5.4 for $k := 2d \log n$. The partitioning of Π's rectangles is done more carefully: We sort all original rectangles in *reverse topological order* $R_1, R_2, \ldots, R_{n^d}$ from leaves to root, that is, if R_i is a descendant of R_j then R_i comes before R_j in the order. Then we process the rectangles in this order:

Initialize cumulative error sets $X^*_{\mathsf{err}} = Y^*_{\mathsf{err}} := \emptyset$. *Iterate for* $i = 1, 2, \ldots, n^d$ rounds:

1. Remove from R_i the rows/columns X^*_{err}, Y^*_{err}. That is, update
$$R_i \;\leftarrow\; R_i \smallsetminus \big(X^*_{\mathsf{err}} \times \{0,1\}^{mn} \cup [m]^n \times Y^*_{\mathsf{err}}\big).$$
2. Apply the Rectangle Scheme for R_i. Output all resulting subrectangles that satisfy the "structured" case of Lemma 5.4 for $k := 2d \log n$. (All non-structured subrectangles are omitted). Call the resulting error rows/columns X_{err} and Y_{err}.
3. Update $X^*_{\mathsf{err}} \leftarrow X^*_{\mathsf{err}} \cup X_{\mathsf{err}}$ and $Y^*_{\mathsf{err}} \leftarrow Y^*_{\mathsf{err}} \cup Y_{\mathsf{err}}$.

In words, an original rectangle R_i is processed only after all of its descendants are partitioned. Each descendant may contribute some error rows/columns, accumulated into sets X^*_{err}, Y^*_{err}, which are deleted from R_i before it is partitioned. The partitioning of R_i will in turn contribute its error rows/columns to its ancestors.

We may now repeat the proof of Section 5.2.2 verbatim *using only the structured subrectangles output by the above process.* That is, we still maintain the same invariant: when the game state is ρ, we maintain a ρ-structured R' (output by the above process) of an original R. We highlight only the key points below.

(1) Root case. The cumulative error at the end of the process is tiny: X^*_{err}, Y^*_{err} have density at most $n^d \cdot n^{-2d} \leq 1/4$ by a union bound over all rounds. In particular, the root rectangle R_{n^d} (with errors removed) still has density $> 1/2$ inside $[m]^n \times \{0,1\}^{mn}$, and so the partition output by Rectangle Scheme is trivial, containing only the $*^n$-structured R_{n^d} itself. This meets the starting condition for the game.

(2) Internal step. By construction, the cumulative error sets *shrink* when we take a step from a node to one of its children. This means that our error handling does not interfere with the internal step: each structured subrectangle R' of an original rectangle R is wholly covered by the structured subrectangles of R's children.

(3) Leaf case. This case is unchanged.

5.3 Reductions to mKW Search Problems

In this section, we show how to reduce the composed search problems $S \circ g^n$ to monotone KW search problems corresponding to some function $f : \{0,1\}^N \to \{0,1\}$. When combined with Theorem 5.1, this yields monotone circuit lower bounds. We describe two types of reductions.

- **Generic reductions.** We recall a known reduction from total search problems with small non-deterministic communication complexity to mKW search problems. In particular, this allows us to prove Corollary 5.2.
- **Reductions to monotone $\mathcal{C}$-Sat.** While slightly inefficient in terms of number of input variables to f, this approach gives a very clean description of the final function. In particular, this allows us to prove the lower bound on monotone circuits size of 3Xor-Sat (Corollary 5.3).

We remark that the *only* reason that our approach yields only *monotone* circuit lower bounds is because we only know how to reduce to *monotone* KW search problems.

5.3.1 Generic Reductions

Lemma 5.7 (Generic Reduction to mKW). *For all unsatisfiable n-variate k-CNFs F with ℓ clauses, $S(F) \circ \text{Ind}_m^n$ reduces to $\mathsf{mKW}(f)$ for an explicit monotone function $f\colon \{0,1\}^N \to \{0,1\}$, where $N \leq \ell \cdot (2m)^k$.*

A rectangle $R \subseteq \mathcal{X} \times \mathcal{Y}$ is said to be *monochromatic* for a search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ if $R \subseteq S^{-1}(o)$ for some $o \in \mathcal{O}$. The nondeterministic communication complexity of S is the logarithm of the least number of monochromatic rectangles that cover the whole input domain

$\mathcal{X} \times \mathcal{Y}$. It is easy to see that $\mathsf{mKW}(f)$ for any monotone $f : \{0,1\}^N \to \{0,1\}$ has nondeterministic communication complexity of $\log N$, with one rectangle to cover all inputs $(x, y) \in \mathsf{mKW}(f)^{-1}(i)$ for each $i \in [N]$. The following lemma establishes a converse (we provide a proof for completeness).

Lemma 5.8 (cf. [Gál01, Lemma 2.3])**.** *If S has nondeterministic communication complexity $\log N$, then S reduces to $\mathsf{mKW}(f)$ for some monotone $f\colon \{0,1\}^N \to \{0,1\}$.*

Proof. Let $\{R_1, \ldots, R_N\}$ be the set of monochromatic rectangles that cover the entire domain $\mathcal{X} \times \mathcal{Y}$, where $R_i = X_i \times Y_i$. Define maps $A : \mathcal{X} \to \{0,1\}^N$ and $B : \mathcal{Y} \to \{0,1\}^N$ such that,

$$A(x)_i = \mathbf{1}\{x \in X_i\} \qquad \text{and} \qquad B(y)_i = \mathbf{1}\{y \notin Y_i\}$$

In words, $A(x)$ is the indicator vector of the subset of rectangles among the cover, that the row corresponding to x intersects with, and $B(y)$ is the *complement* of such an indicator vector for the column corresponding to y. For any (x, y), there exists a coordinate $i \in [N]$ such that $(x, y) \in R_i$ and hence $A(x)_i = 1$ and $B(y)_i = 0$. Thus, we can define a (partial) monotone function $f : \{0,1\}^N \to \{0,1\}$ where $f(A(x)) = 1$ for all $x \in \mathcal{X}$ and $f(B(y)) = 0$ for all $y \in \mathcal{Y}$. Thus, S reduces to $\mathsf{mKW}(f)$ as any solution $i \in [N]$ to $\mathsf{mKW}(f)$ points us to a monochromatic rectangle R_i and hence a solution to S. □

Proof of Lemma 5.7. Consider a composed search problem $S(F) \circ g^n$ obtained from a k-CNF contradiction with ℓ clauses. Its nondeterministic communication complexity is at most $\log \ell + k \cdot (\log m + 1)$; it takes $\log \ell$ bits to specify an unsatisfied clause C, and $\log m + 1$ bits to verify the output of a single gadget, and there are k gadgets relevant to C. □

Proof of Corollary 5.2. This follows readily from Lemma 5.7 combined with Theorem 5.1 and the equivalence of monotone circuits for f and rectangle-dags solving $\mathsf{mKW}(f)$ (Lemma 4.4), namely,

$$\text{monotone circuit size}(f) \;=\; \mathsf{rect\text{-}dag}(\mathsf{mKW}(f)) \;\geq\; \mathsf{rect\text{-}dag}(S(F) \circ \textsc{Ind}_m^n) \;\geq\; n^{\Omega(w)}. \qquad \square$$

5.3.2 Reductions to Monotone $\mathcal{C}$-Sat

$\mathcal{C}$-Sat. Fix an alphabet Σ (potentially infinite, e.g., $\Sigma = \mathbb{R}$). Let $\mathcal{C}$ be a finite set of k-ary predicates over Σ, that is, each $C \in \mathcal{C}$ is a function $C\colon \Sigma^k \to \{0,1\}$. We define a monotone function $\mathcal{C}\text{-}\textsc{Sat}_n\colon \{0,1\}^N \to \{0,1\}$ over $N = |\mathcal{C}|n^k$ input bits as follows. An input $x \in \{0,1\}^N$ is interpreted as a $\mathcal{C}$-CSP instance, that is, x is (the indicator vector of) a set of $\mathcal{C}$-constraints, each applied to a k-tuple of variables from $v_1, \ldots, v_n$. We define $\mathcal{C}\text{-}\textsc{Sat}_n(x) \coloneqq 1$ iff the $\mathcal{C}$-CSP x is *unsatisfiable*: no assignment $v \in \Sigma^n$ exists such that $C(v) = 1$ for all $C \in x$.

For a field $\mathbb{F}$, we define $k\textsc{Lin}(\mathbb{F})$ as the set of all $\mathbb{F}$-linear equations of the form

$$\textstyle\sum_{i\in[k]} a_i v_i \;=\; a_0, \qquad \text{where } \; a_i \in \{0, \pm 1\}.$$

In particular, we recover $3\textsc{Xor}\text{-}\textsc{Sat}_n$ (as in Corollary 5.3) essentially as $3\textsc{Lin}(\mathbb{F}_2)\text{-}\textsc{Sat}_n$. We could have allowed the a_i to range over $\mathbb{F}$ when $\mathbb{F}$ is finite, but we stick with the above convention as it ensures that the set $k\textsc{Lin}(\mathbb{R})$ is always finite.

Boolean alphabets. We assume henceforth that all alphabets Σ contain distinguished elements 0 and 1. We define $\mathcal{C}_{\text{bool}}$ to be the constraint set obtained from $\mathcal{C}$ by restricting each $C \in \mathcal{C}$ to the boolean domain $\{0,1\}^k \subseteq \Sigma^k$. Moreover, if F is a $\mathcal{C}$-CSP, we write F_{bool} for the $\mathcal{C}_{\text{bool}}$-CSP obtained by restricting the constraints of F to boolean domains. Consequently, any $S(F_{\text{bool}})$ associated with a $\mathcal{C}$-CSP F is a *boolean* search problem.

We show that a lifted version of $S(F_{\text{bool}})$, where F is an unsatisfiable $\mathcal{C}$-CSP, reduces to the monotone Karchmer–Wigderson game for $\mathcal{C}$-Sat. Note that we require F to be unsatisfiable over its original alphabet Σ, but the reduction is from the booleanized (and hence easier-to-refute) version of F.

Lemma 5.9. *Let F be an unsatisfiable $\mathcal{C}$-CSP. Then $S(F_{\text{bool}}) \circ \textsc{Ind}^n_m$ reduces to $\mathsf{mKW}(\mathcal{C}\text{-}\textsc{Sat}_{nm})$.*

Proof. Suppose the $\mathcal{C}$-CSP F consists of constraints $C_1, \ldots, C_t$ applied to variables $z_1, \ldots, z_n$. We reduce $S(F_{\text{bool}}) \circ \textsc{Ind}^n_m \subseteq [m]^n \times (\{0,1\}^m)^n \times [t]$ to the problem $\mathsf{mKW}(f) \subseteq f^{-1}(1) \times f^{-1}(0) \times [N]$

where $f := \mathcal{C}\text{-}\textsc{Sat}_{mn}$ over $N := |\mathcal{C}|(mn)^k$ input bits. The two parties compute locally as follows.

Akbar: Given $(x_1, \ldots, x_n) \in [m]^n$, Akbar constructs a $\mathcal{C}$-CSP over variables $\{v_{i,j} : (i,j) \in [n] \times [m]\}$ that is obtained from F by renaming its variables $z_1, \ldots, z_n$ to $v_{1,x_1}, \ldots, v_{n,x_n}$ (in this order). Since F was unsatisfiable, so is Akbar's variable-renamed version of it. Thus, when interpreted as an indicator vector of constraints, Akbar has constructed a 1-input of $\mathcal{C}\text{-}\textsc{Sat}_{mn}$.

Birbal: Given $y \in (\{0,1\}^m)^n$, Birbal constructs a $\mathcal{C}$-CSP over variables $\{v_{i,j} : (i,j) \in [n] \times [m]\}$ as follows. We view y naturally as a boolean assignment to the variables $v_{i,j}$. Birbal includes in his $\mathcal{C}$-CSP instance all possible $\mathcal{C}$-constraints C applied to the $v_{i,j}$ such that C is satisfied under the assignment y (i.e., $C(y) = 1$). This is clearly a satisfiable $\mathcal{C}$-CSP instance, as the assignment y satisfies all Birbal's constraints. Thus, when interpreted as an indicator vector of constraints, Birbal has constructed a 0-input of $\mathcal{C}\text{-}\textsc{Sat}_{mn}$.

It remains to argue that any solution to $\mathsf{mKW}(\mathcal{C}\text{-}\textsc{Sat}_{mn})$ gives rise to a solution to $S(F_{\text{bool}}) \circ \textsc{Ind}_m^n$. Indeed, a solution to $\mathsf{mKW}(\mathcal{C}\text{-}\textsc{Sat}_{mn})$ corresponds to a $\mathcal{C}$-constraint C that is present in Akbar's $\mathcal{C}$-CSP but not in Birbal's. By Birbal's construction, such a C must be violated by the assignment y (i.e., $C(y) = 0$). Since all Akbar's constraints involve only variables $v_{1,x_1}, \ldots, v_{n,x_n}$, the constraint C must in fact be violated by the partial assignment to the said variables, which is $z = \textsc{Ind}_m^n(x,y)$. Thus the constraint of F from which C was obtained via renaming is naturally associated to a solution to $S(F_{\text{bool}}) \circ \textsc{Ind}_m^n$. □

Proof of Corollary 5.3. By Lemma 4.4, it suffices to show

$$\mathsf{rect\text{-}dag}(\mathsf{mKW}(3\textsc{Xor}\text{-}\textsc{Sat}_n)) \;\geq\; \exp(n^{\Omega(1)}).$$

Urquhart [Urq87] exhibited unsatisfiable n-variate 3Xor-CSPs F (aka *Tseitin formulas*) requiring linear Resolution width, that is, $w(S(F)) \geq \Omega(n)$ in our notation. Hence Theorem 5.1 implies that $\mathsf{rect\text{-}dag}(S(F) \circ \textsc{Ind}_m^n) \geq \exp(\Omega(n))$ for $m = n^{\Delta}$. By the reduction in Lemma 5.9, we get that $\mathsf{rect\text{-}dag}(\mathsf{mKW}(3\textsc{Xor}\text{-}\textsc{Sat}_{nm})) \geq \exp(\Omega(n))$. (Note that 3Xor has a boolean alphabet, so $F = F_{\text{bool}}$.) This yields the desired lower bound by reparameterizing the number of variables. □

In fact, we can further generalize Corollary 5.3 to a lower bound for $3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n$. In order to do so, we first state a resolution width lower bound for a $k\textsc{Lin}(\mathbb{F})$-CSP.

Lemma 5.10. *For any field $\mathbb{F} \in \{\mathbb{F}_p : \text{prime } p\} \cup \{\mathbb{R}\}$ and large enough constant k, there exists an $k\textsc{Lin}(\mathbb{F})$-CSP F over n Boolean variables with $O(n)$ constraints that has resolution width $\Omega(n)$.*

Proof. Fix such an $\mathbb{F}$ henceforth. We start with a $k\textsc{Lin}(\mathbb{F})$-CSP introduced in [BGIP01] for $\mathbb{F} = \mathbb{F}_p$ (aka *mod-p Tseitin formulas*), but the definition generalizes to any field. The CSP is constructed based on a given directed graph $G = (V, E)$ that is *regular*: $\text{in-deg}(v) = \text{out-deg}(v) = k/2$ for all $v \in V$. Fix also a distinguished vertex $v^* \in V$. Then $F = F_{G,\mathbb{F}}$ is defined as the following $k\textsc{Lin}(\mathbb{F})$-CSP over variables $\{z_e : e \in E\}$:

$$\forall v \in V : \quad \sum_{(v,u)\in E} z_{(v,u)} - \sum_{(u,v)\in E} z_{(u,v)} = \mathbb{1}_{v^*}(v), \tag{$F_{G,\mathbb{F}}$}$$

where $\mathbb{1}_{v^*}(v^*) = 1$ and $\mathbb{1}_{v^*}(v) = 0$ for $v \neq v^*$. This system is unsatisfiable because the sum over $v \in V$ of the RHS equals 1 whereas the sum of the LHS equals 0 (each variable appears once with a positive sign, once with a negative sign).

We claim that the Booleanized k-CSP F_{bool} (more precisely, its natural k-CNF encoding — cf. Remark 2.4) has linear Resolution width, that is $w(S(F_{\text{bool}})) \geq \Omega(n)$. Indeed, the constraints of F_{bool} are $k/2$-*robust* in the sense that if a partial assignment $\rho \in \{0, 1, *\}^k$ fixes the value of a constraint of F_{bool}, then ρ must set more than $k/2$ variables. Alekhnovich et al. [ABRW04, Theorem 3.1] show that if k is a large enough constant, there exist regular expander graphs G such that F_{bool} (or any k-CSP with $\Omega(k)$-robust constraints) has Resolution width $\Omega(n)$, as desired. □

Remark 5.11. The natural k-CNF encoding of $F_{G,\mathbb{F}}$ considered in Lemma 5.10 has $\mathsf{NS}_{\mathbb{F}}$-refutations of degree $O(k)$, since each linear constraint can be derived from its corresponding CNF constraints with degree $O(k)$ and summing up all linear constraints gives us 1. We will use this observation later in Chapter 6, to give a separation between $\mathsf{NS}_{\mathbb{F}}$ and Cutting Planes refutations.

Corollary 5.12. *For any field $\mathbb{F} \in \{\mathbb{F}_p : \text{prime } p\} \cup \{\mathbb{R}\}$,*

$$3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n \text{ requires monotone circuits of size } 2^{n^{\Omega(1)}}.$$

Proof. By Lemma 4.4, it suffices to show that,

$$\mathsf{rect\text{-}dag}(\mathsf{mKW}(3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n)) \geq 2^{n^{\Omega(1)}}.$$

We already get $\mathsf{rect\text{-}dag}(\mathsf{mKW}(k\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n)) \geq 2^{n^{\Omega(1)}}$ for large enough k, by using Lemma 5.10 with Theorem 5.1 and the reduction in Lemma 5.9 (analogous to the proof of Corollary 5.3). We can reduce the arity from k to 3 by a standard trick. For example, given the linear constraint $a_1v_1 + a_2v_2 + a_3v_3 + a_4v_4 = a_0$ we can introduce a new auxiliary variable u and two equations $a_1v_1 + a_2v_2 + u = 0$ and $-u + a_3v_3 + a_4v_4 = a_0$. In general, we can create an equisatisfiable $3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}$ instance by replacing each equation on $k > 3$ variables with a collection of $k - 2$ equations by introducing $k - 3$ auxiliary variables for each possible $k\textsc{Lin}(\mathbb{F})$ constraint. This shows that $k\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n$ is a monotone projection of $3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_{kn^{O(k)}}$, thus giving us the desired lower bound on $\mathsf{rect\text{-}dag}(\mathsf{mKW}(3\textsc{Lin}(\mathbb{F})\text{-}\textsc{Sat}_n))$ by reparameterizing the number of variables. □

Chapter 6

Cutting Planes Lower Bounds

In this chapter we significantly strengthen Theorem 5.1. We prove a characterization of the triangle-dag complexity for *composed* search problems of the form $S \circ g^n$, in terms of the conjunction-dag width of S. This shows that all the inequalities in Equation 4.1 (in Chapter 4) are tight for such composed search problems.

Theorem 6.1. *Let $m = m(n) \coloneqq n^{\Delta}$ for $\Delta \geq 20$. For any $S \subseteq \{0,1\}^n \times \mathcal{O}$,*

$$\textsf{tri-dag}(S \circ \textsc{Ind}_m^n) \;=\; n^{\Theta(w(S))}.$$

As applications, we get lower bounds against monotone real circuits and Cutting Planes refutations.

Monotone Real Circuits. Following similar reductions as in Chapter 5, we can get that the lower bounds in Corollaries 5.2, 5.3 and 5.12 also hold against monotone *real* circuits (with slightly worse constants in the exponent).

Cutting Planes. Another pithy corollary is that if we start with any CNF contradiction F that is hard for Resolution and compose F with a gadget, the formula becomes hard for Cutting Planes. In fact, the composed CNF can itself be encoded such that each clause has width at most $3k$.

Corollary 6.2. *Any unsatisfiable k-CNF formula F on n variables that requires Resolution width w to refute, is associated with an unsatisfiable $3k$-CNF F' on $n^{O(1)}$ variables, such that any Cutting Planes refutation for F' has length at least $n^{\Omega(w)}$.*

Thus, starting with an unsatisfiable CNF F which has low-degree Nullstellensatz refutations (cf. Remark 5.11), but requires large Resolution width to refute, we can use the above corollary to show that Nullstellensatz refutations can be exponentially more powerful than log of the minimum length of Cutting Planes refutations.

Corollary 6.3. *For any field $\mathbb{F} \in \{\mathbb{F}_p : \text{ prime } p\} \cup \{\mathbb{R}\}$, there exists an unsatisfiable CNF F over n variables, which is refuted by an $\mathbb{F}$-Nullstellensatz proof of $O(\log n)$ degree, but requires Cutting Planes refutations of length $2^{n^{\Omega(1)}}$.*

Previously, only few examples of hard contradictions were known for Cutting Planes, all proved via feasible interpolation [Pud97, HC99, HP17, FPPR17]. A widely-asked question has been to improve this state-of-the-art by developing alternative lower bound methods; see the surveys [BP01, §4] and [Raz16, §5]. In particular, Jukna [Juk12, Research Problem 19.17] asked to find a more intuitive "combinatorial" proof method "explicitly showing what properties of [contradictions] force long derivations." While our lower bound does implicitly use the feasible interpolation method for Cutting Planes, at least it does afford a simple intuition: the hardness is simply borrowed from the realm of Resolution (where we understand very well what makes formulas hard).

6.1 Lifting for Triangle-Dags

In this section we prove the nontrivial direction of Theorem 6.1: Let Π be a triangle-dag solving $S \circ G$ of size n^d for some d. Our goal is to show that $w(S) \leq O(d)$.

The proof is conceptually the same as for rectangle-dags. The only difference is that we need to replace Rectangle Scheme (and the associated Lemma 5.4) with an algorithm that partitions a given triangle $T \subseteq [m]^n \times \{0,1\}^{mn}$ into structured subtriangles that behave like conjunctions.

6.1.1 Triangle partition scheme

We introduce a triangle partitioning algorithm, Triangle Scheme. Its precise definition is postponed to Section 6.2. For now, we only need its high-level description: On input a triangle T, Triangle Scheme outputs a disjoint cover $\bigsqcup_i R^i \supseteq T$ where R^i are rectangles. This induces a partition of

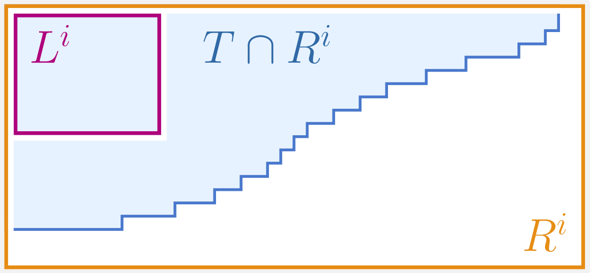


Figure 6.1: Structured case of Lemma 6.4: The subtriangle $T \cap R^i$ is sandwiched between two ρ^i-structured rectangles L^i and R^i.

T into subtriangles $T \cap R^i$. Each (non-error) rectangle R^i is ρ^i-structured (for low-width ρ^i) and is associated with a ρ^i-structured "inner" subrectangle $L^i \subseteq R^i$ satisfying $L^i \subseteq T \cap R^i \subseteq R^i$; see Figure 6.1. Hence $T \cap R^i$ is ρ^i-like, as it is sandwiched between two ρ^i-like rectangles.

More formally, all the properties of Triangle Scheme that we will subsequently need are formalized below (note the similarity with Lemma 5.4); see Section 6.2.2 for the proof.

Lemma 6.4 (Triangle Lemma). *Fix any parameter $k \leq n \log n$. Given a triangle $T \subseteq [m]^n \times \{0,1\}^{mn}$, let $\bigsqcup_i R^i \supseteq T$ be the output of Triangle Scheme. Then there exist "error" sets $X_{\mathsf{err}} \subseteq [m]^n$ and $Y_{\mathsf{err}} \subseteq \{0,1\}^{mn}$, both of density $\leq 2^{-k}$ (inside their respective sets), such that for each i, one of the following holds:*

- ***Structured case:*** *R^i is ρ^i-structured for some ρ^i of width at most $O(k/\log n)$. Moreover, there exists an "inner" rectangle $L^i \subseteq T \cap R^i$ such that L^i is also ρ^i-structured.*
- ***Error case:*** *R^i is covered by error rows/columns, i.e., $R^i \subseteq X_{\mathsf{err}} \times \{0,1\}^{mn} \cup [m]^n \times Y_{\mathsf{err}}$.*

*A **query alignment** property holds: for every $x \in [m]^n \smallsetminus X_{\mathsf{err}}$, there exists a subset $I_x \subseteq [n]$ with $|I_x| \leq O(k/\log n)$ such that every "structured" R^i intersecting $\{x\} \times \{0,1\}^{mn}$ has $\mathsf{fix}\, \rho^i \subseteq I_x$.*

6.1.2 Simplified proof

As in the rectangle case, we first give a simplified proof assuming no errors. That is, invoking Triangle Scheme for each triangle T involved in Π, we assume that

(†) *Assumption:* All rectangles in the cover $\bigsqcup_i R^i \supseteq T$ output by Triangle Scheme satisfy the "structured" case of Lemma 6.4 for $k := 2d \log n$.

The argument for getting rid of the assumption (†) is the same as in the rectangle case, and hence we omit that step—one only needs to observe that removing cumulative error rows/columns from a triangle still leaves us with a triangle.

Overview. As before, we extract a width-$O(d)$ Explorer-strategy for S by walking down the triangle-dag Π, starting at the root. For each triangle T of Π that is reached in the walk, we maintain a ρ-structured inner rectangle $L \subseteq T$. Here ρ (of width $O(d)$ by the choice of k) will record the current state of the game. There are the three steps (1)–(3) to address, of which (1) and (3) remain exactly the same as in the rectangle case. So we only explain step (2), which requires us to replace the use of Lemma 5.4 with the new Lemma 6.4.

(2) Internal step. Supposing the game has reached state ρ_L and we are maintaining some ρ_L-structured inner rectangle $L \subseteq T$ associated with an internal node v, we want to move to some $\rho_{\widetilde{L}}$-structured inner rectangle $\widetilde{L} \subseteq \widetilde{T}$ associated with a child of v. Moreover, we must keep the width of the game state at most $O(d)$ during this move.

Since $L =: X' \times Y'$ is ρ_L-structured, we have from Lemma 3.5 that there exists some $x^* \in X'$ such that $\{x^*\} \times Y'$ is ρ_L-like. Let the two triangles associated with the children of v be T_0 and T_1, so that $L \subseteq T_0 \cup T_1$.

Let $\bigsqcup_i R_b^i$ be the rectangle cover of T_b output by Triangle Scheme. By query alignment in Lemma 6.4, there is some $I_b^* \subseteq [n]$, $|I_b^*| \leq O(d)$, such that all R_b^i that intersect the x^*-th row are ρ^i-structured with $\mathsf{fix}\,\rho^i \subseteq I_b^*$. As Explorer, we now query the input variables in coordinates $J := (I_0^* \cup I_1^*) \smallsetminus \mathsf{fix}\,\rho_L$ (in any order) obtaining some response string $z_J \in \{0,1\}^J$ from the Adversary. As a result, the state of the game becomes the extension of ρ_L by z_J, call it ρ^*, which has width $|\mathsf{fix}\,\rho^*| = |\mathsf{fix}\,\rho_L \cup J| \leq O(d)$.

Note that there is some $y^* \in Y'$ (and hence $(x^*, y^*) \in L \subseteq T_0 \cup T_1$) such that $G(x^*, y^*)$ is consistent with ρ^*; indeed, the whole row $\{x^*\} \times Y'$ is ρ_L-like and ρ^* extends ρ_L. Suppose $(x^*, y^*) \in T_0$; the case of T_1 is analogous. In the rectangle covering of T_0, let R be the unique part such that $(x^*, y^*) \in R$. Note that R is ρ_R-like for some ρ_R that is consistent with $G(x^*, y^*)$ and $\mathsf{fix}\,\rho_R \subseteq I_0^*$ (by query alignment). Hence ρ^* extends ρ_R. As Explorer, we now forget all queried

Triangle Scheme

Input: Triangle $T \subseteq [m]^n \times \{0,1\}^{mn}$ with labeling functions (a_T, b_T)
Output: A disjoint rectangle cover $\bigsqcup_i R^i \supseteq T$

```
$Y_{\mathsf{err}} \leftarrow$ Column Cleanup on $T$
Initialize $\mathcal{R}^0_{\text{alive}} := \{[m]^n \times (\{0,1\}^{mn} \setminus Y_{\mathsf{err}})\}$;   $\mathcal{R}^r_{\text{alive}} := \emptyset$ for all $r \geq 1$;   $\mathcal{R}_{\text{final}} := \emptyset$
loop for $r = 0, 1, 2, \ldots,$ rounds until $\mathcal{R}^r_{\text{alive}}$ is empty:
    for all $R \in \mathcal{R}^r_{\text{alive}}$ do
        $\bigsqcup_i R^i \leftarrow$ Rectangle Scheme on $R$ relative to free coordinates
        for all parts $R^i$ do
            if $|X^{T \cap R^i}| \geq |X^{R^i}|/2$ then
                Add $R^i$ to $\mathcal{R}_{\text{final}}$
            else
                $R^{i,\text{top}} :=$ top half of $R^i$ according to $a_T$  (in particular $T \cap R^i \subseteq R^{i,\text{top}}$)
                Add $R^{i,\text{top}}$ to $\mathcal{R}^{r+1}_{\text{alive}}$ subject to $T \cap R^{i,\text{top}} \neq \emptyset$
return $\mathcal{R}_{\text{final}} \cup \{[m]^n \times Y_{\mathsf{err}}\}$
```

variables in ρ^* except those queried in ρ_R. Also we move to the inner rectangle $\widetilde{L} \subseteq R$ promised by Lemma 6.4 that satisfies $\widetilde{L} \subseteq T_0$ and is $\rho_{\widetilde{L}} = \rho_R$ structured.

We have recovered our invariant: the game state is $\rho_{\widetilde{L}}$ and we maintain a $\rho_{\widetilde{L}}$-structured subrectangle $\widetilde{L}$ of a triangle T_0. Moreover, the width of the game state remained $O(d)$.

6.2 Triangle Partitioning Scheme

In the description of Triangle Scheme, we denote projections of a set $S \subseteq [m]^n \times \{0,1\}^{mn}$ by

$$\begin{aligned} X^S &:= \{x \in [m]^n : \exists y \in \{0,1\}^{mn} \text{ such that } (x,y) \in S\}\,, \\ Y^S &:= \{y \in \{0,1\}^{mn} : \exists x \in [m]^n \text{ such that } (x,y) \in S\}\,. \end{aligned}$$

Overview. Triangle Scheme computes a disjoint rectangle cover $\bigsqcup_i R^i$ of T. Starting with a trivial cover of the whole communication domain by a single part, the algorithm progressively *refines* this cover over several rounds as guided by the input triangle T. As outlined in Section 6.1.1, the goal is to end up with ρ-structured rectangles R^i that contain a large enough portion of T so that we may sandwich $L^i \subseteq T \cap R^i \subseteq R^i$ where L^i is a ρ-structured "inner" rectangle.

The main idea is as follows. The algorithm maintains a pool of *alive* rectangles. In a single round,

Column Clean-up

Input: Triangle $T \subseteq [m]^n \times \{0,1\}^{mn}$ with labeling functions (a_T, b_T)
Output: Error columns $Y_{\mathsf{err}} \subseteq \{0,1\}^{mn}$

```
$Y_{\mathsf{err}} \leftarrow \emptyset$
For $I \subseteq [n]$, $\alpha \in [m]^I$, $\gamma \in \{0,1\}^I$, define $Y_{I,\alpha,\gamma} := \{y \in \{0,1\}^{mn} : g^I(\alpha, y_I) = \gamma\}$
while there exists $I, \alpha, \gamma, x$ such that $0 < |T \cap (\{x\} \times (Y_{I,\alpha,\gamma} \smallsetminus Y_{\mathsf{err}}))| < 2^{mn-n^2}$ do
    $Y_{\mathsf{err}} \leftarrow Y_{\mathsf{err}} \cup Y^{T \cap (\{x\} \times Y_{I,\alpha,\gamma})}$
return $Y_{\mathsf{err}}$
```

for each alive rectangle R, we first invoke Rectangle Scheme in order to restore ρ-structuredness for the resulting subrectangles R^i. Then for each R^i we check if the subtriangle $T \cap R^i$ occupies at least half the rows of R^i. If *yes*, we add it to the *final* pool, which will eventually form the output of the algorithm. If *no*, we discard the "lower" half of R^i as determined by the labeling a_T, that is, the half that does not intersect T. The "top" half (containing $T \cap R^i$) will enter the alive pool for next round.

Column Cleanup. An important detail is the subroutine Column Cleanup, run at the start of Triangle Scheme, which computes a small set of columns that will eventually be declared as Y_{err}. By discarding the columns Y_{err}, we ensure that whatever subrectangle R^i is output by Rectangle Scheme, the rows of $T \cap R^i$ will satisfy an *empty-or-heavy dichotomy*: for every $x \in X^{R^i}$, the x-th row of $T \cap R^i$ is either empty, or "heavy", that is, of size at least 2^{mn-n^2}. Having many heavy rows helps towards satisfying the 3rd item in Definition 3.3 of ρ-stucturedness, and hence in finding the inner rectangle L^i. This property of Column Cleanup is formalized in Claim 6.5 below.

Free coordinates. Another detail to explain is the underlined phrase *relative to free coordinates*. For each alive rectangle R we tacitly associate a subset of *free coordinates* $J_R \subseteq [n]$ and *fixed coordinates* $[n] \smallsetminus J_R$. At start, the single alive rectangle has $J_R := [n]$, and whenever we invoke Rectangle Scheme for a rectangle R *relative to free coordinates*, the understanding is that in line (i) of Rectangle Scheme, the choice of I_i is made among subsets of J_R alone. The resulting subrectangle $R^i = X^i \times Y^i$, obtained by fixing the coordinates I_i in X^i, will have its free coordinates $J_{R^i} := J_R \smallsetminus I_i$. (Restricting a rectangle to its top half on line 10 does not modify the free coordinates.)

6.2.1 Properties of Triangle Scheme

Claim 6.5. *For a triangle $T \subseteq [m]^n \times \{0,1\}^{mn}$, let Y_{err} be the output of Column Cleanup. Then:*

- Empty-or-heavy: *For every triple $(I \subseteq [n], \alpha \in [m]^I, \gamma \in \{0,1\}^I)$, and every $x \in [m]^n$, it holds that $T \cap (\{x\} \times (Y_{I,\alpha,\gamma} \setminus Y_{\mathsf{err}}))$ is either empty or has size at least 2^{mn-n^2}.*
- Size bound: $|Y_{\mathsf{err}}| \leq 2^{mn-\Omega(n^2)}$.

Proof. The first property is immediate by definition of Column Cleanup. For the second property, in each while-iteration, at most 2^{mn-n^2} columns get added to Y_{err}. Moreover, there are no more than $2^n \cdot m^n \cdot 2^n \cdot m^n = (2m)^{2n}$ choices of $I \subseteq [n]$, $\alpha \in [m]^I$, $\gamma \in \{0,1\}^I$ and $x \in [m]^n$, and the loop executes at most once for each choice of I, α, γ, x. Thus, $|Y_{\mathsf{err}}| \leq (2m)^{2n} \cdot 2^{mn-n^2} \leq 2^{mn-\Omega(n^2)}$. □

Next, we list some key invariants that hold for Triangle Scheme.

Lemma 6.6. *For every $r \geq 0$, there exists a partition $\mathcal{X}^r := \{X^i\}_i$ of $[m]^n$ satisfying the following.*

(P1) *For every $R \in \mathcal{R}^r_{\text{alive}}$ we have $X^R \in \mathcal{X}^r$.*

(P2) *Each $X^i \in \mathcal{X}^r$ is labeled by a pair $(I_i \subseteq [n], \alpha_i \in [m]^{I_i})$ such that $\boldsymbol{X}^i_{I_i} = \alpha_i$ is fixed.*

(P3) *The partition $\mathcal{X}^{r+1}$ is a refinement of $\mathcal{X}^r$. The labels respect this: if $X^j \in \mathcal{X}^{r+1}$ is a subset of $X^i \in \mathcal{X}^r$, then $I_j \supseteq I_i$ and α_j agrees with α_i on coordinates I_i.*

Moreover, let $\mathcal{X} := \mathcal{X}^{r^*}$ be the final partition assuming Triangle Scheme completes in r^* rounds.

(P4) *For every $R \in \mathcal{R}_{\text{final}}$ the row set X^R is a union of parts of $\mathcal{X}$. If $X^i \in \mathcal{X}$, labeled (I_i, α_i), is such that $X^R \supseteq X^i$, then the fixed coordinates of R are a subset of I_i.*

(P5) *For every $r \geq 0$, $\mathcal{X}^r$ and $\mathcal{X}$ agree on a fraction $\geq 1 - 2^{-r}$ of rows, that is, there is a subset of "final" parts $\mathcal{X}^r_{\text{final}} \subseteq \mathcal{X}^r$ such that $\bigcup \mathcal{X}^r_{\text{final}}$ has density $\geq 1 - 2^{-r}$ inside $[m]^n$, and $\mathcal{X}^r_{\text{final}} \subseteq \mathcal{X}$.*

Proof. Let us define the row partitions $\mathcal{X}^r$. The partition $\mathcal{X}^1$ contains only a single part, $[m]^n$, labeled by $I_1 := \emptyset$. Supposing $\mathcal{X}^r$ has been defined, the next partition $\mathcal{X}^{r+1}$ is obtained by refining each old part $X^i \in \mathcal{X}^r$. Consider one such old part $X^i \in \mathcal{X}^r$ with label (I_i, α_i). If there is *no* rectangle $R \in \mathcal{R}^r_{\text{alive}}$ with $X^R = X^i$ then we need not partition X^i any further; we simply include X^i in $\mathcal{X}^{r+1}$ as a whole. Otherwise, let $R \in \mathcal{R}^r_{\text{alive}}$ be any rectangle such that $X^R = X^i$; we emphasize that there can be many such choices for R, but the upcoming refinement of X^i will not depend on that choice. The r-th round of the algorithm first computes $R = \bigsqcup_i R^i$ using Rectangle Scheme, and then each R^i might be horizontally split in half. We interpret this as a refinement of X^i according to the 1st round of Rectangle Scheme on R (which only depends on $X^R = X^i$), with each part adding more fixed coordinates to the label (I_i, α_i). Letting $X^i = \bigsqcup_j X^{i,j}$ denote the resulting row partition, we then split each $X^{i,j}$ into two halves $X^{i,j,\mathrm{top}}$ and $X^{i,j,\mathrm{bot}}$. This completes the definition of $\mathcal{X}^{r+1}$.

The properties (P1)–(P5) are straightforward to verify. For (P5), we only note that when the algorithm horizontally splits a rectangle (inducing $X^{i,j} = X^{i,j,\mathrm{top}} \cup X^{i,j,\mathrm{bot}}$), the bottom halves are discarded, and never again touched in future rounds. That is, $X^{i,j,\mathrm{bot}} \in \mathcal{X}^{r'}$ for all $r' > r$. This cuts the number of "alive" rows $\bigcup_{R \in \mathcal{R}^r_{\mathrm{alive}}} X^R$ in half each round. □

Lemma 6.7 (Error rows)**.** *Let $\mathcal{X} = \{X^i\}_i$ be the final row partition in Lemma 6.6. Fix any parameter $k < n \log n$. There is a density-2^{-k} subset $X_{\mathsf{err}} \subseteq [m]^n$ (which is a union of parts of $\mathcal{X}$) such that for any part $X^i \not\subseteq X_{\mathsf{err}}$, we have $|I_i| \leq O(k/\log n)$.*

Proof. Our strategy is as follows (cf. [GPW17, Lemma 7]). For $x \in [m]^n$, let $i(x)$ be the unique index such that $x \in X^{i(x)} \in \mathcal{X}$; recall that $X^{i(x)}$ is labeled by some $(I_{i(x)}, \alpha_{i(x)})$. We will study a uniform random $\boldsymbol{x} \sim [m]^n$ and show that the distribution of the number of fixed coordinates $|I_{i(\boldsymbol{x})}|$ has an exponentially decaying tail. This allows us to define X_{err} as the set of outcomes of $\boldsymbol{x}$ for which $|I_{i(\boldsymbol{x})}|$ is exceptionally large. More quantitatively, it suffices to show for a large constant C,

$$\Pr\left[\, |I_{i(\boldsymbol{x})}| > C \cdot k/\log n \,\right] \;\leq\; 2^{-k}. \tag{6.1}$$

Recall that $\mathcal{X}$ and $\mathcal{X}^\ell$, where $\ell := k+1$, agree on all but a fraction $2^{-k}/2$ of rows by (P5). Hence by a union bound, it suffices to show a version of Eq. 6.1 truncated at level ℓ:

$$\Pr\left[\, |I_{i'(\boldsymbol{x})}| > C' \cdot \ell/\log n \,\right] \;\leq\; 2^{-\ell} \quad (= 2^{-k}/2), \tag{6.2}$$

where $i'(x)$ is defined as the unique index with $x \in X^{i'(x)} \in \mathcal{X}^\ell$.

Partitions as a tree. The sequence $\mathcal{X}^0, \ldots, \mathcal{X}^\ell$, of row partitions can be visualized as a depth-ℓ

tree where the nodes at depth r corresponds to parts of $\mathcal{X}^r$, and there is an edge from $X \in \mathcal{X}^r$ to $X' \in \mathcal{X}^{r+1}$ iff $X' \subseteq X$. A way to generate a uniform random $\boldsymbol{x} \sim [m]^n$ is to take a random walk down this tree, starting at the root:

- At a non-leaf node $X \in \mathcal{X}^r$ we take a tree edge (X, X') with probability $|X'|/|X|$.
- Once at a leaf node $X \in \mathcal{X}^\ell$, we output a uniform random $\boldsymbol{x} \sim X$.

Potential function. We define a nonnegative potential function on the nodes of the tree. For each part $X \in \mathcal{X}^r$, labeled $(I \subseteq [n], \alpha \in \{0,1\}^I)$, we define

$$D(X) \;\coloneqq\; (n - |I|)\log m - \log|X| \;\geq\; 0.$$

How does the potential change as we take a step starting at node $X \in \mathcal{X}^r$ labeled (J, α)? If X has one child, the value of D remains unchanged. Otherwise, we move to a child of X in two substeps.

- *Substep 1:* Recall that we partition $X = \bigsqcup_i X^i$ according to the 1st round of Rectangle Scheme relative to free coordinates. That is, X^i is further restricted on $I_i \subseteq [n] \smallsetminus J$ to some value $\alpha_i \in [m]^{I_i}$. For a child X^i labeled $(J \sqcup I_i, \alpha \sqcup \alpha_i)$ the potential change is

$$\begin{aligned} D(X^i) - D(X) &= (n - |J \cup I_i|)\log m - \log|X^i| - (n - |J|)\log m + \log|X| \\ &= \log|X| - \log|X^i| - |I_i|\log m \\ &= \log(|X|/|X^{\geqslant i}|) - \log(|X^i|/|X^{\geqslant i}|) - |I_i|\log m \\ &= \log(|X|/|X^{\geqslant i}|) - \log\Pr[\boldsymbol{X}^{\geqslant i}_{I_i} = \alpha_i] - |I_i|\log m \\ &\leq \log(|X|/|X^{\geqslant i}|) + 0.95|I_i|\log m - |I_i|\log m \\ &= \delta(i) - 0.05|I_i|\log m. \qquad\qquad (\text{where } \delta(i) \coloneqq \log(|X|/|X^{\geqslant i}|)) \end{aligned}$$

- *Substep 2:* Each X^i gets split into two halves, $X^{i,\mathrm{top}}$ and $X^{i,\mathrm{bot}}$. Moving to either child makes the potential increase by exactly 1 bit.

In summary, when we take a step to a random child in our random walk, the overall change in potential is itself a random variable, which is at most

$$\boldsymbol{\delta} - 0.05|\boldsymbol{I}|\log m + 1, \tag{6.3}$$

where $(\boldsymbol{I}, \cdot)$ is the label of the random child, and $\boldsymbol{\delta} := \delta(\boldsymbol{i})$ is the random variable generated by choosing $\boldsymbol{i}$ with $\Pr[\boldsymbol{i} = i] = |X^i|/|X|$. Summing Eq. 6.3 over ℓ many rounds, we see that ℓ steps of the random walk takes us to a node $X^{\boldsymbol{j}} \in \mathcal{X}^\ell$ with random index $\boldsymbol{j}$, which is labeled $(I_{\boldsymbol{j}}, \alpha_{\boldsymbol{j}})$, and which satisfies $D(X^{\boldsymbol{j}}) \le \sum_{r\in[\ell]}(\boldsymbol{\delta}_r + 1) - 0.05|I_{\boldsymbol{j}}|\log m$ where $\boldsymbol{\delta}_r$ is the "$\boldsymbol{\delta}$" variable corresponding to the r-th step. Since the potential is nonnegative, we get that

$$|I_{\boldsymbol{j}}| \;\le\; \frac{20}{\log m} \cdot \sum_{r\in[\ell]}(\boldsymbol{\delta}_r + 1). \tag{6.4}$$

Bounding this quantity is awkward since, in general, the variables $\boldsymbol{\delta}_r$ are not mutually independent. However, a standard trick to overcome this is to define mutually independent and identically distributed random variables $\boldsymbol{d}_r$ and couple them with $\boldsymbol{\delta}_r$ so that $\boldsymbol{\delta}_r \le \boldsymbol{d}_r$ with probability 1.

- *Definition of $\boldsymbol{d}_r$:* Sample a uniform real $\boldsymbol{p}_r \in [0, 1)$ and define $\boldsymbol{d}_r := \log(1/(1 - \boldsymbol{p}_r))$ and couple with $\boldsymbol{\delta}_r$ such that $\boldsymbol{\delta}_r = \delta(\boldsymbol{i})$ where $\boldsymbol{i}$ is such that $\boldsymbol{p}_r$ falls in the $\boldsymbol{i}$-th interval, assuming we have partitioned $[0, 1)$ into half-open intervals with lengths $|X^i|/|X|$ (where $X^1, X^2, \ldots$ are the sets from Substep 1) in the natural left-to-right order. Now $\boldsymbol{\delta}_r$ is correctly distributed and $\boldsymbol{\delta}_r \le \boldsymbol{d}_r$ with probability 1.

Note that $\mathbb{E}[2^{\boldsymbol{d}_r/2}] = \int_0^1 1/(1-p)^{1/2}\mathrm{d}p = 2$. For a large enough constant $C > 0$, we calculate

$$\begin{aligned}
\Pr\left[\, \textstyle\sum_{r\in[\ell]} \boldsymbol{d}_r > C\ell \,\right] \;&=\; \Pr[2^{\sum_{r\in[\ell]}(\boldsymbol{d}_r/2)} > 2^{C\ell/2}] \\
&\le\; \mathbb{E}[2^{\sum_{r\in[\ell]}(\boldsymbol{d}_r/2)}]/2^{C\ell/2} \\
&=\; \left(\textstyle\prod_{r\in[\ell]} \mathbb{E}[2^{\boldsymbol{d}_r/2}]\right)/2^{C\ell/2} \\
&=\; 2^\ell/2^{-C\ell/2} \;\le\; 2^{-C\ell/3}.
\end{aligned}$$

Plugging this estimate in Eq. 6.4 (using $\boldsymbol{\delta}_r \le \boldsymbol{d}_r$) we get that $\Pr[|I_{\boldsymbol{j}}| > C' \cdot \ell/\log n] < 2^{-\ell}$ for a sufficiently large C'. This proves Eq. 6.2 and concludes the proof of the lemma. □

6.2.2 Proof of Triangle Lemma (Lemma 6.4)

Identifying Y_{err}*,* X_{err}*.* The column error set Y_{err} is already defined by Triangle Scheme. Note that only one rectangle, $[m]^n \times Y_{\mathsf{err}}$, is covered by the error columns. Claim 6.5 ensures that Y_{err} has density at most $2^{-\Omega(n^2)} < 2^{-k}$. The row error set X^{err} is defined by Lemma 6.7 (for the given k).

Structured vs. error. Let $\bigsqcup_i R^i$ be the output of Triangle Scheme, and consider an $R^i = X^i \times Y^i$ which is not covered by error rows/columns; in particular $R^i \in \mathcal{R}_{\text{final}}$. Let $I_i \subseteq [n]$ denote the fixed coordinates of R^i such that $\boldsymbol{X}^i_{I_i} = \alpha_i$ for some $\alpha_i \in \{0,1\}^{I_i}$. From Claim 5.6 we have that $\boldsymbol{X}^i_{[n]\smallsetminus I_i}$ is 0.95-dense. From (P4) and Lemma 6.7 we have $|I_i| \leq O(k/\log n)$. Moreover, we observe that $Y^i = Y_{I_i,\alpha_i,\gamma_i} \smallsetminus Y_{\mathsf{err}}$ for some $\gamma_i \in \{0,1\}^{I_i}$ (notation from Column Cleanup) since Rectangle Scheme, and hence Triangle Scheme by extension, only partitions columns by fixing individual gadget outputs. We have $|Y_{I_i,\alpha_i,\gamma_i}| \geq 2^{mn-n}$ by definition, and so $|Y^i| \geq 2^{mn-2n}$ is large enough: we conclude that R^i is ρ^i-structured for ρ^i that equals γ_i on I_i and consists of stars otherwise.

Next, we locate the associated inner rectangle $L^i \subseteq R^i$. All final rectangles output by Triangle Scheme are such that $|X^{(T\cap R^i)}| \geq |X^i|/2$. That is, every top row in $R^{i,\text{top}}$ has a nonempty intersection with T. Hence the empty-vs-heavy property of Claim 6.5 says that for all $x \in X^{i,\text{top}}$, we have $|T \cap (\{x\} \times Y^i)| \geq 2^{mn-n^2}$. Moreover, note that $\boldsymbol{X}^{i,\text{top}}$ is 0.9-dense on its free coordinates $[n] \smallsetminus I_i$ (we lose at most 1 bit of min-entropy compared to $\boldsymbol{X}^i$ by Fact 5.5). We can now define $L^i := X^{i,\text{top}} \times Y' \subseteq T \cap R^i$ where Y' is the set of the first (according to b_T) 2^{mn-n^2} columns of Y^i; see Figure 6.1. This L^i meets all the conditions for being ρ^i-structured.

Query alignment. For $x \in [m]^n \smallsetminus X_{\mathsf{err}}$, we define (I_x, α_x) as the label of the unique part $i(x)$ such that $x \in X^{i(x)} \in \mathcal{X}$. By Lemma 6.7, $|I_x| \leq O(k/\log n)$. Every ρ-structured rectangle $R^j := X^j \times Y^j$ with $X^j \supseteq X^{i(x)}$ is, by (P4), such that $\mathsf{fix}\,\rho \subseteq I_x$.

6.3 Reductions to CNF Search Problems

In this section, we show how to reduce the composed search problems $S(F) \circ g^n$ to back to a CNF search problem $S(F')$ in a generic fashion, thereby proving Corollaries 6.2 and 6.3.

Lemma 6.8 (Generic Reduction to CNF Search)**.** *For all unsatisfiable n-variate k-CNFs F with ℓ clauses, $S(F) \circ \textsc{Ind}_m^n$ reduces to $S(F')$ (with a certain partition of variables) for an explicit unsatisfiable $2mn$-variate $3k$-CNF F' with $\ell \cdot m^k$ clauses.*

The key property of an n-variable search problem $S \subseteq \{0,1\}^n \times \mathcal{O}$ that facilitates an efficient reduction to a CNF search problem is having a low *certificate* (a.k.a. nondeterministic query) complexity. A *certificate for* $(z, o) \in S$ is a partial assignment $\rho \in \{0,1,*\}^n$ such that z is consistent with ρ and o is a valid output for every input consistent with ρ; in short, $z \in C_\rho^{-1}(1) \subseteq S^{-1}(o)$. A *certificate for z* is a certificate for $(z, o) \in S$ for some $o \in S(x)$. The *certificate complexity of z* is the least width of a certificate for z. The *certificate complexity of S* is the maximum over all $z \in \{0,1\}^n$ of the certificate complexity of z.

It is easy to see that $S(F)$ for any k-CNF contradiction $F = \bigwedge_i D_i$ has certificate complexity at most k, with a width-k certificate ρ such that $C_\rho = \neg D_i$ that certifies all $z \in D_i^{-1}(0) = S(F)^{-1}(i)$ for each i. The following lemma establishes a converse.

Lemma 6.9 (cf. [LNNW95])**.** *If $S \subseteq \{0,1\}^n \times \mathcal{O}$ has nondeterministic query complexity k, then S reduces to $S(F)$ for some k-CNF n-variate contradiction F.*

Proof. We can pick a collection $\mathcal{C}$ of width-k certificates, one for each $z \in \{0,1\}^n$. The k-CNF formula F is then defined as $\bigwedge_{\rho \in \mathcal{C}} \neg C_\rho$. Any solution to $S(F)$ points us to a certificate ρ of S and hence a solution to S. □

For the purposes of query complexity, there are two ways to represent the first argument $x \in [m]$ to the index function $\textsc{Ind}_m \colon [m] \times \{0,1\}^m \to \{0,1\}$ as a binary string. The simplest is to write x as a binary string in $\{0,1\}^{\log m}$. Under this convention, $\textsc{Ind}_m$ has certificate complexity $\log m + 1$. If $S \subseteq \{0,1\}^n \times \mathcal{O}$ has certificate complexity k, the composed problem $S \circ \textsc{Ind}_m^n$ has certificate complexity $k(\log m + 1)$ (by composing certificates). For applications, this means that if we start with a k-CNF contradiction F, we may reduce $S(F) \circ \textsc{Ind}_m^n$ to solving $S(F')$ where F' is a $k(\log m + 1)$-CNF contradiction over $O(mn)$ variables.

A better representation [BHP10, dRNV16], which does not blow up the certificate complexity (or CNF width), is to write x as an m-bit string of Hamming weight 1 (the index of the unique

1-entry encodes $x \in [m]$). Under this convention, $\textsc{Ind}_m \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ becomes a *partial* function of certificate complexity 2. Hence, if S has certificate complexity k, the *partial* composed problem $S' \coloneqq S \circ \textsc{Ind}_m^n$ has certificate complexity $2k$. Also note that under this encoding the query complexity of the gadget blows up to $m+1$. We would like a *total* search problem, preferably without blowing up the query complexity of each gadget.

Proof of Lemma 6.8. We slightly change the encoding of x without making it any easier to solve for any of the dag models. Specifically, for inputs $x \in (\{0,1\}^m)^n$ and $y \in (\{0,1\}^m)^n$ to Akbar and Birbal, we will say that $z \in \{0,1\}^n$ is consistent with (x, y) if for each $i \in [n]$, there is an index $k \in [m]$ such that $x_{i,k-1} = 0$, $x_{i,k} = 1$ and $y_{i,k} = z_i$, where we let $x_{i,0} = 0$ by convention. We define our total search problem $S_{\text{tot}} \subseteq \{0,1\}^{mn} \times \{0,1\}^{mn} \times \mathcal{O}$ via all the width-$3k$ certificates that certify that some z consistent with (x, y) violates some clause of F; there are a total of $\ell \cdot m^k$ such certificates. We have that S_{tot} is at least as hard as $S(F) \circ \textsc{Ind}_m^n$ for any of the dag models, since Akbar can map $x' \in [m]^n$ to an $x \in (\{0,1\}^m)^n$ such that for each i, $x_{i,1} = \ldots = x_{i,k-1} = 0$ and $x_{i,k} = \ldots = x_{i,m} = 1$ for $k = x'_i$. This ensures that the only z consistent with (x, y) is precisely $z = \textsc{Ind}_m^n(x', y)$. Thus, any output $o \in S_{\text{tot}}(x, y)$ is such that $o \in S(F)(x', y)$.

Hence, by Lemma 6.9, we can reduce (in the context of communication) $S(F) \circ \textsc{Ind}_m^n$ to solving $S(F')$ where F' is a $3k$-CNF contradiction over $2mn$ variables and $\ell \cdot m^k$ clauses. □

Proof of Corollary 6.2. Starting with n-variate k-CNF F, we reduce to N-variate $3k$-CNF F' as in Lemma 6.8 (for $N = 2mn = 2n^{\Delta+1}$), which along with the bipartition of the input variables to Akbar and Birbal we interpret as a communication search problem F''. We have,

$$\begin{aligned}
\text{Cutting Planes length of } F' &= \mathsf{thresh\text{-}dag}(S(F')) \\
&\geq \mathsf{tri\text{-}dag}(S(F'')) && \ldots(\text{from Eq. 4.1}) \\
&\geq \mathsf{tri\text{-}dag}(S(F) \circ \textsc{Ind}_m^n) && \ldots(\text{by Lemma 6.8}) \\
&\geq n^{\Omega(w(S(F)))} && \ldots(\text{by Theorem 6.1}) \quad \square
\end{aligned}$$

Proof of Corollary 6.3. For any field $\mathbb{F}$, we have from Lemma 5.10 and Remark 5.11, that there exists an n-variate k-CNF $F = F_{G,\mathbb{F}}$ that requires resolution width of $\Omega(n)$, but has $\mathsf{NS}_{\mathbb{F}}$-refutations of degree $O(k)$. In fact, $k = O(1)$.

From Corollary 6.2, we have that the associated F', obtained via reduction from $S(F) \circ \textsc{Ind}_m^n$ to $S(F')$, requires Cutting Planes refutations of length $2^{n^{\Omega(1)}}$.

Finally, observe that $\mathsf{NS}_{\mathbb{F}}$ behaves well under the composition as described in the proof of Lemma 6.8. Firstly, observe that for the composition described, there is always at least one z consistent with (x, y) which can be obtained via binary search on x. Thus, it takes only $\log m + 1 = O(\log n)$ (since $m = n^{\Delta}$) queries to find a canonical z consistent with (x, y). Thus, we can write each variable z_i of this canonical z as a degree $O(\log n)$ polynomial in the variables (x_i, y_i). Thus, $\mathsf{NS}_{\mathbb{F}}(F') \leq \mathsf{NS}_{\mathbb{F}}(F) \cdot O(\log n) \leq O(\log n)$. □

Chapter 7

Monotone Span Program Lower Bounds

In this chapter, we prove an exponential lower bound on the size of monotone $\mathbb{F}_p$-span programs for computing a function computable by linear sized $\mathbb{R}$-span programs.

Theorem 7.1. *For any prime p, $3\textsc{Lin}(\mathbb{R})\text{-}\textsc{Sat}_n$ requires monotone $\mathbb{F}_p$-span programs of size $2^{n^{\Omega(1)}}$.*

Pitassi and Robere [PR18] have already exhibited functions (in hindsight, $3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_n$) computable by linear sized monotone $\mathbb{F}_p$ span programs, but requires exponential sized monotone $\mathbb{R}$-span programs, and also monotone $\mathbb{F}_q$-span programs (for prime $q \neq p$). Our result proves a separation in the other direction.

Theorem 7.1 is unrelated to the lifting theorems proved in Chapters 5 and 6. Instead we use a lifting theorem, proved by Pitassi and Robere [PR18], that characterizes the monotone $\mathbb{F}$-span program size of certain functions in terms of the $\mathbb{F}$-Nullstellensatz degree of a corresponding unsatisfiable CNF. Thus, we first provide an unsatisfiable CSP that requires large $\mathsf{NS}_{\mathbb{F}_p}$-degree refutations, but has small $\mathsf{NS}_{\mathbb{R}}$-degree refutations and then lift this result to monotone span programs.

7.1 Nullstellensatz Lower Bounds

Following notations in Section 5.3.2, we will show the following.

Lemma 7.2. *For any $\mathbb{F} \in \{\mathbb{F}_p : \text{prime } p\} \cup \{\mathbb{R}\}$, there exists an unsatisfiable* $3\textsc{Lin}(\mathbb{R})$*-CSP F such that $\mathsf{NS}_{\mathbb{F}_p}(F_{\text{bool}}) \geq n^{\Omega(1)}$.*

To this end, we consider an $\mathbb{R}$-linear system $F = F_{G,U,\mathbb{R}}$ that generalizes $F_{G,\mathbb{R}}$ defined in the context of Lemma 5.10:

$$\forall v \in V: \quad \sum_{(v,u)\in E} z_{(v,u)} - \sum_{(u,v)\in E} z_{(u,v)} = \mathbb{1}_U(v), \tag{$F_{G,U,\mathbb{R}}$}$$

where $\mathbb{1}_U \colon V \to \{0,1\}$ is the indicator function for $U \subseteq V$. This is unsatisfiable as long as $U \neq \emptyset$. Combinatorially, the CSP Search Problem $S(F_{\text{bool}})$ can be interpreted as an $\textsc{End-of-}\ell\textsc{-Lines}$ problem for $\ell := |U|$: given a directed graph (encoded by the $z_{(v,u)}$ variables) with distinguished source vertices U, find a sink or a source not in U. It is important to have many distinguished sources, $|U| \geq n^{\Omega(1)}$, as otherwise F_{bool} has small $\mathsf{NS}_{\mathbb{Z}}$-degree refutations[1] and hence F_{bool} has low $\mathbb{F}_p$-Nullstellensatz degree refutations as well.

To show $\mathsf{NS}_{\mathbb{F}_p}(F_{\text{bool}}) \geq n^{\Omega(1)}$ for an suitable choice of $F = F_{G,U,\mathbb{R}}$, we adapt a result of Beame and Riis [BR98]. They proved an $\mathbb{F}_p$-Nullstellensatz degree lower bound for a related *bijective pigeonhole* principle P_n whose underlying graph has *unbounded* degree; we obtain a bounded-degree version of their result by a reduction.

Lemma 7.3 ([BR98, §8])**.** *Fix a prime p. The following system of (unsatisfiable) polynomial equations over variables $\{x_{ij} : (i,j) \in D \times R\}$, where $|D| = n$ and $|R| = n - n^{\Omega(1)}$, requires $\mathbb{F}_p$-Nullstellensatz degree $n^{\Omega(1)}$:*

$$\begin{array}{llrcll}
\textit{(i)} & \forall i \in D: & \sum_{j\in R} x_{ij} & = & 1 & \textit{“each pigeon occupies a hole”,} \\
\textit{(ii)} & \forall j \in R: & \sum_{i\in D} x_{ij} & = & 1 & \textit{“each hole houses a pigeon”,} \\
\textit{(iii)} & \forall i \in D, \{j,j'\} \in \binom{R}{2}: & x_{ij}x_{ij'} & = & 0 & \textit{“no pigeon occupies two holes”,} \\
\textit{(iv)} & \forall j \in R, \{i,i'\} \in \binom{D}{2}: & x_{ij}x_{i'j} & = & 0 & \textit{“no hole houses two pigeons”,} \\
\textit{(v)} & \forall (i,j) \in D \times R: & x_{ij}^2 - x_{ij} & = & 0 & \textit{“Boolean axioms”.}
\end{array} \tag{P_n}$$

[1] jumping a little ahead, one way to derive this is to use a result of Hollender and Goldberg [HG18] that $\textsc{Multi-Source-End-of-Line}$ is in $\mathsf{PPAD}^{\mathsf{dt}}$ and combine it with the observation that PPAD–decision trees yield low $\mathsf{NS}_{\mathbb{Z}}$-degree refutations (Theorem 8.19).

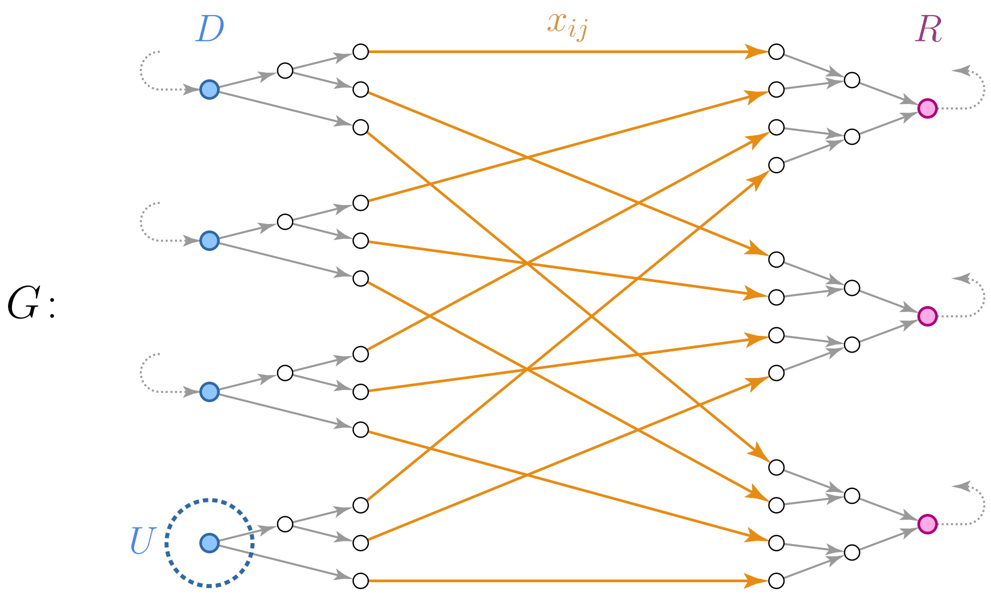


Figure 7.1: Graph $G = (V, E)$, a bounded-degree version of the biclique $D \times R$.

We construct a natural bounded-degree version G of the complete bipartite graph $D \times R$ and show that each constraint of F_{bool} for $F = F_{G,U,\mathbb{R}}$ is a low-degree $\mathbb{F}_p$-Nullstellensatz consequence of P_n. Hence, if F_{bool} admits a low-degree $\mathbb{F}_p$-Nullstellensatz proof, so does P_n (see, e.g., [BGIP01, Lemma 1] for composing proofs), which contradicts Lemma 7.3.

The directed graph $G = (V, E)$ is obtained from the complete bipartite graph $D \times R$ as illustrated in Figure 7.1 (for $|D| = 4$ and $|R| = 3$). Specifically, each vertex of degree d in $D \times R$ is replaced with a binary tree of height $\log d$. The result is a layered graph with the first and last layers identified with D and R, respectively. We also add a "feedback" edge from each vertex in R to a vertex in D according to some arbitrary injection $R \to D$ (dashed edges in Figure 7.1). The vertices in D not incident to feedback edges will form the set U (singleton in Figure 7.1).

This defines a Boolean 3-CSP F_{bool} for $F = F_{G,U,\mathbb{R}}$ over variables $\{z_e : e \in E\}$. In order to reduce P_n to F_{bool}, we define an affine map between the variables x_{ij} of P_n and z_e of F_{bool}. Namely, for a feedback edge e we set $z_e \coloneqq 1$, and for every other $e = (v, u)$ we set

$$\begin{aligned} z_{(v,u)} &\coloneqq \sum_{i\in D_v\ j\in R_u} x_{ij}, \\ \text{where} \quad D_v &\coloneqq \{i \in D : v \text{ is reachable from } i \text{ without using feedback edges}\}, \\ R_u &\coloneqq \{j \in R : j \text{ is reachable from } u \text{ without using feedback edges}\}. \end{aligned}$$

Note in particular that this map naturally identifies the edge-variables z_e in the middle of G (yellow edges) with the variables x_{ij} of P_n. The other variables z_e are simply affinely dependent on the middle edge-layer. We then show that from the equations of P_n we can derive each constraint of F_{bool}. Recall that the constraint for $v \in V$ requires that the *out-flow* $\sum_{(v,u)\in E} z_{(v,u)}$ equals the *in-flow* $\sum_{(u,v)\in E} z_{(u,v)}$ (plus 1 iff $v \in U$).

$v \notin D \cup R$: Suppose v is on the left side of G (right side is handled similarly) so that $z_{(v,u)} = \sum_{j\in R_u} x_{ij}$ for some fixed $i \in D$. The out-flow is

$$\sum_{u\,:\,(v,u)\in E} z_{(v,u)} = \sum_{u\,:\,(v,u)\in E}\ \sum_{j\in R_u} x_{ij} = \sum_{j\in R_v} x_{ij}. \tag{7.1}$$

On the other hand, v has a unique incoming edge (u^*, v) so the in-flow is given by $\sum_{u\,:\,(u,v)\in E} z_{(u,v)} = z_{(u^*,v)} = \sum_{j\in R_v} x_{ij}$, which equals Eq. 7.1.

$v \in D$: (Case $v \in R$ is similar). The in-flow equals 1 (either $v \in U$ so that we have the $+1$ term from $\mathbb{1}_U(v)$; or $v \notin U$ and the value of a feedback-edge variable gives $+1$). The out-flow equals $\sum_{j\in R_v} x_{ij} = \sum_{j\in R} x_{ij} = 1$ by Eq. 7.1, $R_v = R$, and axiom (i) of P_n.

Finally, we can verify the boolean axioms $z_e^2 = z_e$. This holds trivially for feedback edges e. Let $e = (v, u)$ be an edge in the left side of G (right side is similar) so that $z_e = \sum_{j\in R_u} x_{ij}$ for some fixed $i \in D$. We have $z_e^2 = (\sum_{j\in R_u} x_{ij})^2 = \sum_{j\in R_u} x_{ij}^2 = \sum_{j\in R_u} x_{ij} = z_e$ by (iii) and (v) axioms of P_n. This concludes the proof of Lemma 7.2.

7.2 Lifting Nullstellensatz to Monotone Span Programs

(Monotone) Span Programs are characterized by the communication model of *Algebraic Partitions* that solve the corresponding (monotone) Karchmer Wigderson search problems. This is analogous to characterization of circuits by rectangle-dags (cf. Lemma 4.4).

Fix a two-party search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$. We say that a matrix $M \in \mathbb{F}^{\mathcal{X}\times\mathcal{Y}}$ is *monochromatic* if there is some $o \in \mathcal{O}$ such that $o \in S(x, y)$ for all (x, y) in the support of M, i.e. $M(x, y) \neq 0$. For any field $\mathbb{F}$, an $\mathbb{F}$-*partition* of a search problem S is a set $\mathcal{M}$ of rank-1 matrices $M \in \mathbb{F}^{\mathcal{X}\times\mathcal{Y}}$ such that $\sum_{M\in\mathcal{M}} M = \mathbb{1}$ and each $M \in \mathcal{M}$ is monochromatic for S. The *size* of the partition is $|\mathcal{M}|$. The $\mathbb{F}$-*partition number* $\chi_{\mathbb{F}}(S)$ is the least size of an $\mathbb{F}$-partition of S. Let $\mathsf{SP}_{\mathbb{F}}$ (and $\mathsf{mSP}_{\mathbb{F}}$) denote the (monotone) span program complexity.

Lemma 7.4 ([Gál01]). *For any boolean function f and any field $\mathbb{F}$, $\mathsf{SP}_{\mathbb{F}}(f) = \chi_{\mathbb{F}}(\mathsf{KW}(f))$. Furthermore, if f is monotone then $\mathsf{mSP}_{\mathbb{F}}(f) = \chi_{\mathbb{F}}(\mathsf{mKW}(f))$.*

With this connection in place, Pitassi and Robere [PR18] showed a lifting theorem characterizing algebraic $\mathbb{F}$-partition number of a composed search problem $S(F) \circ \textsc{Ind}_m^n$ in terms of $\mathsf{NS}_{\mathbb{F}}(F)$.

Theorem 7.5 ([PR18]). *Let $m \geq n^{\Delta}$ for a sufficiently large Δ. For any unsatisfiable CNF F,*

$$\forall\, \mathbb{F} \in \{\mathbb{F}_p : \textit{prime } p\} \cup \{\mathbb{R}\} : \qquad \chi_{\mathbb{F}}(S(F) \circ \textsc{Ind}_m) \;=\; n^{\Theta(\mathsf{NS}_{\mathbb{F}}(F))}$$

The lifting theorem of Pitassi and Robere [PR18] was originally not stated for the Index gadget, but rather for any "good" gadget on $O(\log n)$ bits of input. However, the above can be derived since any such gadget can be embedded into a $\mathrm{poly}(n)$ sized Index gadget. We state the above version as it makes it convenient to prove Theorem 7.1.

Proof of Theorem 7.1. We start with the 3$\textsc{Lin}(\mathbb{R})$-CSP $F := F_{\mathrm{bool}}$ considered in Lemma 7.2. From

Lemma 5.9, we have that $S(F) \circ \textsc{Ind}_m^n$ reduces to $\mathsf{mKW}(3\textsc{Lin}(\mathbb{R})\text{-}\textsc{Sat}_{mn})$. Thus, we have,

$$\begin{aligned}
\mathsf{mSP}_{\mathbb{F}_p}(3\textsc{Lin}(\mathbb{R})\text{-}\textsc{Sat}_{mn}) &= \chi_{\mathbb{F}_p}(\mathsf{mKW}(3\textsc{Lin}(\mathbb{R})\text{-}\textsc{Sat}_{mn})) && \ldots\text{(from Lemma 7.4)}\\
&\geq \chi_{\mathbb{F}_p}(S(F) \circ \textsc{Ind}_m^n) && \ldots\text{(from Lemma 5.9)}\\
&\geq n^{\Omega(\mathsf{NS}_{\mathbb{F}_p}(F))} && \ldots\text{(by Theorem 7.5)}\\
&\geq 2^{n^{\Omega(1)}} && \ldots\text{(by Lemma 7.2)}
\end{aligned}$$

This gives us the desired lower bound on the monotone $\mathbb{F}_p$-span program size of $3\textsc{Lin}(\mathbb{R})\text{-}\textsc{Sat}_n$ by reparameterizing the number of variables. □

Chapter 8

TFNP in Query & Communication

In this chapter, we describe an intimate connection between the lifting theorems studied in this thesis, and the query and communication analogs of sub-classes of TFNP, the class of all total search problems verifiable in polynomial time. This connection is driven the following key observations:

- ▷ Lemma 6.9 : *CNF search problems $S(F)$ are complete for query-*TFNP.
- ▷ Lemma 5.8 : *Monotone KW search problems* $\mathsf{mKW}(f)$ *are complete for communication-*TFNP.

It turns out that we can (in some cases) view proofs of refutations as "query algorithms". We already saw a glimpse of this in this thesis, namely, that dag-like resolution refutations of F are equivalent to conjunction-dags solving $S(F)$. In the language of TFNP, dag-like resolution can be viewed as the query analog of the class PLS. Additionally, the minimal depth of resolution refutations can be viewed as the query analog of the class FP.

Similarly, it turns out (again, in some cases) that one can view computational models as "communication protocols". We already saw two examples in this thesis:

- – Monotone Formulas computing f are equivalent to communication protocols solving $\mathsf{mKW}(f)$. Tree-like protocols can be viewed as the communication analog of the class FP.
- – Monotone Circuits computing f are equivalent rectangle-dags solving $\mathsf{mKW}(f)$. Rectangle-dags can be viewed as the communication analog of the class PLS.

M	Query	Communication	Reference
FP	Resolution depth	Formula size	Theorem 8.11
PLS	Resolution width	Circuit size	Theorem 8.18
PPA_p	$\mathbb{F}_p$-Nullstellensatz degree	$\mathbb{F}_p$-Span Program size	Theorem 8.12

Table 8.1: Characterizations of query and communication analogs of TFNP subclasses. The colored ones are new characterizations proved in this thesis.

We contribute one more such characterization (see Table 8.1): Communication-PPA captures $\mathbb{F}_2$-span programs and Query-PPA captures $\mathbb{F}_2$-Nullstellensatz. More generally, we consider the class PPA_p (class based on modulo-p arguments) and show that Query/Communication-PPA_p captures $\mathbb{F}_p$-Nullstellensatz/$\mathbb{F}_p$-span programs.

In this chapter, we define the query and communication analogs of several TFNP subclasses (§8.1) and describe these characterizations in more detail (§8.2). With this, we can interpret the lifting theorems discussed in this thesis as lifting theorems for these TFNP sub-classes. This allows us to prove separations between communication analogs of these sub-classes (§8.3).

8.1 TFNP Class Definitions

Megiddo and Papadimitriou [MP91] initiated the study of *total* NP *search problems* (TFNP), that is, search problems that have a solution for every input and where a given solution can be efficiently checked for validity. By now, this theory has flowered into a sprawling jungle of widely-studied complexity classes (such as PLS [JPY88], PPA/PPAD/PPP [Pap94], CLS [DP11]) that serve to classify the complexities of many natural search problems. The classes are defined based on the existential mathematical principle that ensures totality of the corresponding search problem. See [GP18a, SZZ18] for detailed surveys.

Any complexity class defined in the language of Turing machines can be analogously defined in query complexity (cf. [Ver99]) and in communication complexity (cf. [BFS86]). It is well known that separations between query complexity classes imply oracle separations between the corresponding Turing machine classes, and separations between communication complexity classes imply *algebraic oracle* separations between the corresponding Turing machine classes [AW09].

Every TFNP subclass in literature is defined as a total search problem on an (exponentially large) *object* (such as a graph) that is succinctly encoded by the input, typically interpretted as a circuit. For each TFNP subclass, there is a canonical definition of its communication or query analog: we simply let communication protocols or decision trees (rather than circuits) implicitly define the objects that appear in the original Turing machine definition.

Each communication class M^{cc} (resp. query class M^{dt}) is defined via a M*–protocol* (resp. M–*decision tree*) that solves a two-party search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ (resp. $S \subseteq \{0,1\}^n \times \mathcal{O}$). We use $\mathsf{M}^{\mathsf{cc}}(S)$ to denote the least cost of an M–protocol that solves S (analogously for $\mathsf{M}^{\mathsf{dt}}(S)$). With a slight abuse of notation, the class M^{cc} (resp. M^{dt}) is then defined as the set of all search problems S that have $\mathsf{M}^{\mathsf{cc}}(S) \leq \text{poly}\log(\log(|\mathcal{X}| \cdot |\mathcal{Y}|))$ (resp. $\mathsf{M}^{\mathsf{dt}}(S) \leq \text{polylog}(n)$).

Below, we define the communication analogs of various TFNP-subclasses with the understanding that a query version can be obtained by replacing mentions of a protocol $\Pi_v(x, y)$ by a decision tree $\mathcal{T}_v(z)$; the *cost* of a M–decision tree is defined as $\max_v |\mathcal{T}_v|$ where $|\mathcal{T}_v| \coloneqq \max_z \#(\text{queries made by } \mathcal{T}_v(z))$. In what follows, *sink* means a vertex with out-degree 0 and in-degree > 0, and *source* means a vertex with in-degree 0 and out-degree > 0. We also define *weak sink* to mean a vertex with out-degree 0 (but in-degree may be 0).

Definition 8.1. (FP – Search problems solvable in Polynomial time)

Syntax: Π is a (deterministic) protocol outputting values in $\mathcal{O}$.

Correctness: $\Pi(x, y) \in S(x, y)$.

Cost: $|\Pi| \coloneqq$ communication cost of Π.

Definition 8.2. (TFNP – Search problems solvable in Non-deterministic Polynomial time)

Syntax: Π is a non-deterministic protocol, on each path either aborts or outputs value in $\mathcal{O}$.

Correctness: $o \in S(x, y)$, for output o on any non-aborting path of $\Pi(x, y)$.

Cost: $|\Pi| \coloneqq$ non-deterministic communication cost of Π.

We observed that CNF search problems $S(F)$ are *complete* for $\mathsf{TFNP}^{\mathsf{dt}}$ (Lemma 6.9) and Monotone KW search problems $\mathsf{mKW}(f)$ are *complete* for $\mathsf{TFNP}^{\mathsf{cc}}$ (Lemma 5.8). Thus, the study of TFNP and its subclasses is very natural in our context. We now proceed to define the communication analogs of various TFNP sub-classes (with query analogs defined similarly).

Definition 8.3. (PPA – Polynomial Parity Arguments)

Principle: A matching on an odd sized graph has isolated vertices.

Syntax: V is a vertex set with $|V| \neq 0 \pmod 2$. For each $v \in V$: $o_v \in \mathcal{O}$ and Π_v is a protocol outputting another vertex $\Pi_v(x, y) \in V$.

Object: Matching graph $G_{x,y} = (V, E)$ where $\{v, u\} \in E$ iff $v = \Pi_u(x, y)$ and $u = \Pi_v(x, y)$.

Correctness: If v has degree 0 in $G_{x,y}$, then $o_v \in S(x, y)$.

Cost: $\log |V| + \max_v |\Pi_v|$.

Definition 8.4. (PLS – Polynomial Local Search)

Principle: Every dag has a weak sink.

Syntax: V is a vertex set. For each $v \in V$: $o_v \in \mathcal{O}$ and Π_v is a protocol outputting a pair of successor and potential $(s_v(x, y), \ell_v(x, y)) \in V \times \mathbb{Z}$.

Object: Dag $G_{x,y} = (V, E)$ where $(v, u) \in E$ iff $s_v(x, y) = u$ and $\ell_v(x, y) > \ell_u(x, y)$.

Correctness: If v is a sink in $G_{x,y}$, then $o_v \in S(x, y)$.

Cost: $\log |V| + \max_v |\Pi_v|$.

Definition 8.5. (PPP – Polynomial Pigeonhole Principle)

Principle: Pigeon-hole principle.

Syntax: V is a vertex set with a distinguished vertex $v^* \in V$. For each unordered pair $\{v,u\} \in \binom{V}{2}$: $o_{\{v,u\}} \in \mathcal{O}$. For each $v \in V$: Π_v is a protocol outputting values in $V \smallsetminus \{v^*\}$.

Object: Bipartite graph $G_{x,y} = (V \times (V \smallsetminus \{v^*\}), E)$ where $(v,w) \in E$ iff $\Pi_v(x,y) = w$.

Correctness: If (v,w) and (u,w), $v \neq u$, are edges in $G_{x,y}$, then $o_{\{v,u\}} \in S(x,y)$.

Cost: $\log|V| + \max_v |\Pi_v|$.

Definition 8.6. (PPAD – Polynomial Parity Arguments on Directed graphs)

Principle: Every directed graph with a source must have another source or a sink.

Syntax: V is a vertex set with a distinguished vertex $v^* \in V$. For each $v \in V$: $o_v \in \mathcal{O}$ and Π_v is a protocol outputting a pair of successor and predecessor $(s_v(x,y), p_v(x,y)) \in V \times V$.

Object: Directed graph $G_{x,y} = (V, E)$ where $(v,u) \in E$ iff $s_v(x,y) = u$ and $p_u(x,y) = v$.

Correctness: If v^* is a sink or non-source in $G_{x,y}$, then $o_{v^*} \in S(x,y)$.

If $v \neq v^*$ is a sink or source in $G_{x,y}$, then $o_v \in S(x,y)$.

Cost: $\log|V| + \max_v |\Pi_v|$.

Definition 8.7. (PPADS – Polynomial Parity Arguments on Directed graphs (Sink))

Principle: Every directed graph with a source must have a sink.

Syntax: Same as in Definition 8.6.

Object: Same as in Definition 8.6.

Correctness: If v^* is a sink or non-source in $G_{x,y}$, then $o_{v^*} \in S(x,y)$.

If $v \neq v^*$ is a sink in $G_{x,y}$, then $o_v \in S(x,y)$.

Cost: $\log|V| + \max_v |\Pi_v|$.

We also study PPA_q, which is a *modulo-q* analog of PPA defined, in passing, by Papadimitriou [Pap94]. A thorough study of PPA_q has been done in an ongoing work [GKSZ19], which studies properties of this class in the Turing machine world and gives a natural complete problem for the same.

Definition 8.8. (PPA_q – Polynomial "Parity" Arguments modulo-q)

Principle: A q-dimensional matching on a non-multiple-of-q sized graph has isolated vertices.

Syntax: V is a vertex set with $|V| \neq 0 \,(\mathrm{mod}\ q)$. For each $v \in V$: $o_v \in \mathcal{O}$ and Π_v is a protocol outputting a subset of $q-1$ other vertices $\Pi_v(x,y) \subseteq V$.

Object: q-dim matching $G_{x,y} = (V, E)$ where $\{v_1, \ldots, v_q\} \in E$ iff $v_i \in \Pi_{v_j}(x,y)$ for each $i \neq j$.

Correctness: If v has degree 0 in $G_{x,y}$, then $o_v \in S(x,y)$.

Cost: $\log |V| + \max_v |\Pi_v|$.

While PPA_q can be defined for all integers $q \geq 2$, it is shown in [GKSZ19] that PPA_q can be characterized in terms of PPA_p for all the prime divisors of q. In particular, for any constant q, it is shown that $\mathsf{PPA}_q^{\mathsf{cc}}(S) = \Theta(\min_{p \in P_q} \mathsf{PPA}_p^{\mathsf{cc}}(S))$, where P_q is the set of all prime divisors of q (similarly for the query analogs). Thus, it suffices to study only PPA_p for prime p.

In addition to these well-studied classes, we also define the classes EoPL and SoPL below. EoPL is defined as the class of search problems reducible to End-of-Potential-Line [FGMS18], which in turn is equivalent to End-of-Metered-Line [HY17], a problem in CLS [DP11]. We study EoPL instead of CLS as it is combinatorial and more amenable to defining query/communication analogs. The class SoPL, based on the problem Sink-of-Potential-Line, is newly considered here and is related to EoPL in the same way as PPADS is related to PPAD in that only *sinks* are valid solutions.

Definition 8.9. (EoPL – End of Potential Line)

Principle: Every *potential-ed* directed graph with a source must have another source or a sink.

Syntax: V is a vertex set with a distinguished vertex $v^* \in V$. For each $v \in V$: $o_v \in \mathcal{O}$ and Π_v is a protocol outputting a tuple $(s_v(x,y), p_v(x,y), \ell_v(x,y)) \in V \times V \times \mathbb{Z}$.

Object: Dag $G_{x,y} = (V, E)$ where $(v, u) \in E$ iff $s_v(x,y) = u$, $p_u(x,y) = v$, $\ell_v(x,y) > \ell_u(x,y)$.

Correctness: If v^* is a sink or non-source in $G_{x,y}$, then $o_{v^*} \in S(x,y)$.

If $v \neq v^*$ is a sink or source in $G_{x,y}$, then $o_v \in S(x,y)$.

Cost: $\log |V| + \max_v |\Pi_v|$.

Definition 8.10. (SoPL – Sink of Potential Line)

Principle: Every *potential-ed* directed graph with a source must have a sink.

Syntax: Same as in Definition 8.9.

Object: Same as in Definition 8.9.

Correctness: If v^* is a sink or non-source in $G_{x,y}$, then $o_{v^*} \in S(x, y)$.

If $v \neq v^*$ is a sink in $G_{x,y}$, then $o_v \in S(x, y)$.

Cost: $\log |V| + \max_v |\Pi_v|$.

8.2 Characterizations

The most basic characterization is using query and communication analogs of FP.

Theorem 8.11 (FP, Resolution Depth and Formulas)**.**

1. *For any unsatisfiable CNF F : $\mathsf{FP}^{\mathsf{dt}}(S(F))$ = resolution depth(F).*
2. *For any monotone function f : $\mathsf{FP}^{\mathsf{cc}}(\mathsf{mKW}(f)) = \Theta(\log($monotone formula size$(f)))$.*

   Moreover, for any function f : $\mathsf{FP}^{\mathsf{cc}}(\mathsf{KW}(f)) = \Theta(\log($formula size$(f)))$.

Proof. Part (2.) is precisely Theorem 2.7 [KW88] combined with Lemma 2.8. Part (1.) follows by first considering tree-like resolution (which doesn't increase depth) and observing that top-down version of tree-like resolution is a decision tree (cf. Defns. 4.1, 4.2). □

We describe more such characterizations of proof systems in terms of query analogs, and of computational models in terms of communication analogs of TFNP subclasses. See Figure 8.1 for an illustration of the relative positioning of TFNP subclasses and the characterizations discussed here.

8.2.1 PPA_p, $\mathbb{F}_p$-Nullstellensatz and $\mathbb{F}_p$-Span Programs

In this section, we prove our characterizations for PPA and more generally for PPA_p. Since $\mathsf{PPA} = \mathsf{PPA}_2$, we only state and prove it for PPA_p.

Theorem 8.12 (PPA_p, Nullstellensatz and Span Programs)**.** *For any prime p,*

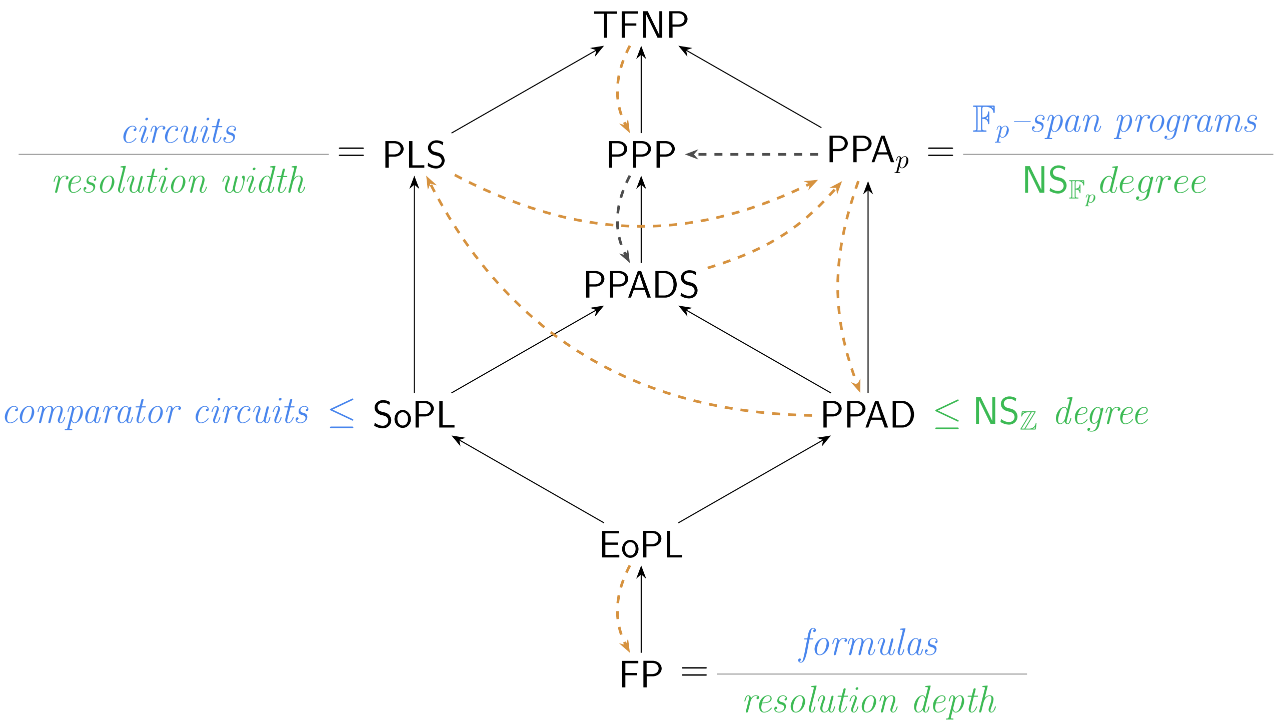

TFNP
circuits
resolution width
= PLS
PPP
PPA$_p$ =
$\mathbb{F}_p$–span programs
NS$_{\mathbb{F}_p}$degree
PPADS
comparator circuits ≤ SoPL
PPAD ≤ NS$_{\mathbb{Z}}$ degree
EoPL
FP =
formulas
resolution depth

Figure 8.1: The landscape of total search problem classes (uncluttered by the usual 'cc', 'dt' superscripts). A solid arrow $\mathsf{M}_1 \to \mathsf{M}_2$ denotes $\mathsf{M}_1 \subseteq \mathsf{M}_2$, and a dashed arrow $\mathsf{M}_1 \dashrightarrow \mathsf{M}_2$ denotes a separation in the query world $\mathsf{M}_1^{\mathsf{dt}} \not\subseteq \mathsf{M}_2^{\mathsf{dt}}$ (in fact, an exponential separation). The yellow arrows are separations that can also be shown in the communication world, namely $\mathsf{M}_1^{\mathsf{cc}} \not\subseteq \mathsf{M}_2^{\mathsf{cc}}$. Communication analogs of some classes characterize models of computation (printed in blue) and Query analogs characterize proof systems (printed in green).

1. *For any unsatisfiable CNF F :* $\mathsf{PPA}_p^{\mathsf{dt}}(S(F)) = \Theta(\mathsf{NS}_{\mathbb{F}_p}(F))$.
2. *For any monotone function f :* $\mathsf{PPA}_p^{\mathsf{cc}}(\mathsf{mKW}(f)) = \Theta(\log(\mathsf{mSP}_{\mathbb{F}_p}(f)))$.

   Moreover, for any function f : $\mathsf{PPA}_p^{\mathsf{cc}}(\mathsf{KW}(f)) = \Theta(\log(\mathsf{SP}_{\mathbb{F}_p}(f)))$.

Communication $\mathsf{PPA}_p = \mathbb{F}_p$-span programs

We first show that communication PPA_p captures $\mathbb{F}_p$-span program size. At a high level, constructing a span program from a PPA_p–protocol is almost immediate from the characterization of span program size (Lemma 7.4) due to Gál [Gál01]. The other direction is more involved and proceeds in two steps: (1) we show that $3\textsc{Lin}(\mathbb{F}_p)$-$\textsc{Sat}_n$ is complete for (monotone) $\mathbb{F}_p$-span programs under (monotone) projections, and then (2) give a PPA–protocol for $3\textsc{Lin}(\mathbb{F}_p)$-$\textsc{Sat}_n$.

Span programs from PPA_p–protocols. To show $\log \mathsf{mSP}_{\mathbb{F}_2}(f) \leq O(\mathsf{PPA}_p^{\mathsf{cc}}(\mathsf{mKW}(f)))$ for a boolean function f, we apply the below lemma with $S \coloneqq \mathsf{mKW}(f)$ and use the characterization $\mathsf{mSP}_{\mathbb{F}_p}(f) = \chi_{\mathbb{F}_p}(\mathsf{mKW}(f))$ in Lemma 7.4. The non-monotone case is similar.

Lemma 8.13. *For any search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ we have $\log \chi_{\mathbb{F}_p}(S) \leq O(\mathsf{PPA}_p^{\mathsf{cc}}(S))$.*

Proof. From a PPA_p–protocol $\Pi \coloneqq (V, o_v, \Pi_v)$ we can obtain a nondeterministic protocol Γ for S. The protocol Γ computes as follows on input (x, y): guess a vertex $v \in V$; if $\deg(v) = 0$ in $G_{x,y}$, then accept (with solution o_v); otherwise reject. In particular, Γ runs $\Pi_v(x, y)$ and then $\Pi_{v_i}(x, y)$ for each vertex $v_i \in \Pi_v(x, y)$. The communication cost is thus at most p times that of Π. By virtue of a PPA_p–protocol, it follows that Γ accepts each input (x, y) a number of times that equals $|V| \pmod p$. By treating each rectangle in the nondeterministic protocol as a matrix containing entries $|V|^{-1} \pmod p$ in its support, we get an $\mathbb{F}_p$-partition for S of size $\exp(O(\mathsf{PPA}_p^{\mathsf{cc}}(S)))$. □

PPA_p–protocols from span programs. As mentioned above, the converse is more involved. We begin by showing that $3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_n$ is complete for $\mathbb{F}_p$-span programs under projections.

Lemma 8.14. *Let f be a (monotone) boolean function computable by a (monotone) $\mathbb{F}_p$-span program of size s. Then f can be written as a (monotone) projection of $3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_{s^2}$.*

Proof. Let M be an $\mathbb{F}_p$-span program for f. We may assume wlog that it is an $\mathbb{F}_p^{s \times s}$ matrix with each row labeled by an input literal, z_i or $\neg z_i$ (or just z_i in the monotone case). By a change of basis we may assume that, instead of the all-1 row vector, the target is to span the row vector $(0, 0, \ldots, 0, 1)$. Let us thus write $M = [A\ b]$ where A is an $s \times (s-1)$ matrix and b is an $s \times 1$ vector. This suggests the following alternative interpretation of the span program M: given an input $z \in \{0, 1\}^n$, accept if and only if the corresponding system of linear equations $A_{(z)} w = b_{(z)}$ consistent with z is *unsatisfiable*; observe that this is witnessed by some linear combination of rows yielding the vector $(0, 0, \ldots, 0, 1)$. This is nearly a projection of $3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}$, except, the number of variables occurring in each linear equation in $Aw = b$ may be greater than 3. This is straightforward to fix by a standard reduction. For example, given the linear constraint $a_1 v_1 + a_2 v_2 + a_3 v_3 + a_4 v_4 = a_0$ we can introduce a new auxiliary variable u and two equations $a_1 v_1 + a_2 v_2 + u = 0$ and $-u + a_3 v_3 + a_4 v_4 = a_0$. In general, we replace each equation on $k > 3$ variables with a collection of $k - 2$ equations by introducing $k-3$ auxiliary variables to create an equisatisfiable instance. The final instance has at most s^2 variables and s^2 equations. □

The following lemma completes the proof that any span program implies a PPA_p–protocol. We prove the lemma only for the monotone game $\mathsf{mKW}(f)$ as it implies the same bound for $\mathsf{KW}(f)$.

Lemma 8.15. $\mathsf{PPA}_p^{\mathsf{cc}}(\mathsf{mKW}(3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_n)) \leq O(\log n)$.

Proof. Write $Av = b$ for the list of all $N := O(n^3)$ many $3\textsc{Lin}(\mathbb{F}_p)$-equations over n variables $v_1, \ldots, v_n$. In the game $\mathsf{mKW}(3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_n)$ Alice holds a subset $x \subseteq [N]$ of the rows of $Av = b$ defining an unsatisfiable system $A_x v = b_x$, and Bob holds a subset $y \subseteq [N]$ defining a satisfiable system $A_y v = b_y$. Their goal is to find an equation which is included in Alice's system but not in Bob's. We fix henceforth some satisfying assignment $w \in \mathbb{F}_p^n$ to Bob's system. It suffices to find an equation in Alice's system that w does not satisfy.

For simplicity, we will assume that Alice's system $A_x v = b_x$ satisfies:

$$\mathbb{1}^T A_x = (0, 0, \ldots, 0) \qquad \text{and} \qquad \mathbb{1}^T b_x = 1 \tag{*}$$

This is indeed without loss of generality: For any unsatisfiable system of equations $\{a_i v = b_i\}_{i \in [m]}$, there exists $\lambda \in \mathbb{F}_p^m$ such that $\sum_{i \in [m]} \lambda_i a_i = (0, 0, \ldots, 0)$ and $\sum_{i \in [m]} \lambda_i b_i = 1$. Alice can discard all equations corresponding to $\lambda_i = 0$. Moreover, for each non-zero λ_i she can multiply the equation $a_i v = b_i$ by λ_i to get a different linear system satisfying the requisite condition above. Alice's system of equations remains unsatisfiable and moreover any equation not satisfied by Bob's w immediately corresponds to an unsatisfied equation in the original system.

The PPA_p–protocol is defined as follows. The vertex set V will contain $(n+1)Np + 1$ vertices. Namely, $(n+1)p$ vertices for each $i \in [N]$ and one additional distinguished vertex v^*. We think of the $(n+1)p$ vertices for each i, as p vertices for each of n variables and p vertices for the constant term. All these $(n+1)p$ vertices are labeled by the corresponding equation.

In order to describe the PPA_p–protocol $\Pi = \{\Pi_v\}_v$, we first describe the p-dimensional matching on any input (x, y) and then turn to describe how such a matching can be encoded via protocols. Assume that i-th equation in $Av = b$ is given by $a_{i,1}v_1 + \ldots + a_{i,n}v_n = a_{i,0}$ (actually, for each i, we have that at most three of the $a_{i,j}$'s are non-zero. But we use this notation for simplicity). We consider hyperedges of three categories as follows:

- ▷ *Non-participating equations:* For each $i \notin x \subseteq [N]$ that does not participate in Alice's system of equations, we simply partition the corresponding $(n+1)p$ vertices into $n+1$ p-uniform hyperedges.
- ▷ *Intra-equations:* For equations $i \in x$ that do participate in Alice's system of equations,
  - *Constant terms:* Among the p vertices correponding to $a_{i,0}$, we reserve $a_{i,0}$ vertices for potential inter-equation hyperedges. From (*), we have $\sum_{i \in x} a_{i,0} = 1$. Thus, in addition to the distinguished vertex, we have totally $0 \pmod{p}$ many vertices left out, which we can partition canonically into p-uniform hyperedges.
  - *Non-constant terms:* Among the p vertices corresponding to $a_{i,j}v_j$, we reserve $a_{i,j}w_j$ vertices for potential inter-equation hyperedges. From (*), for any variable v_j we have that, $\sum_{i \in x} a_{i,j} = 0$. Thus, for any variable v_j we have left out $0 \pmod{p}$ vertices and hence we can partition the remaining vertices into p-uniform hyperedges.
- ▷ *Inter-equations:* For equation $i \in x$, we have $0 \mod p$ vertices left out above iff $\sum_j a_{i,j}w_j = a_{i,0}$. Thus, if equation i is satisfied by Bob's input w, we simply partition the remaining vertices into p-uniform hyperedges. Else, we simply leave them isolated.

It is easy to see that the isolated vertices are precisely the ones corresponding to equations in Alice's system that are unsatisfied by Bob's assignment w. We sketch how such a protocol can be implemented by a $\mathsf{PPA}_p^{\mathsf{cc}}$–protocol of cost $O(\log n)$. Consider a vertex (among p of them) corresponding to the term $a_{i,j}v_j$, with $a_{i,j} \neq 0$ (or even the term $a_{i,0}$). If equation i does not participate in x, then Alice can simply add a p-uniform hyperedge around the p vertices corresponding to $a_{i,j}v_j$ and no communication is required. If equation i does participate in x, Alice sends the variables participating in equation i to Bob and he responds with their values in w. This allows Alice to decide all the inter-equation hyperedges corresponding to the i-th equation and moreover all the intra-equation hyperedges corresponding to variables participating in equation i. In particular, Alice can decide the hyperedge containing the given vertex. □

Query $\mathsf{PPA}_p = \mathbb{F}_p$-Nullstellensatz

We now show that query PPA_p captures $\mathbb{F}_p$-Nullstellensatz degree. Showing that $\mathbb{F}_p$-Nullstellensatz degree lower bounds $\mathsf{PPA}_p^{\mathsf{dt}}$ complexity follows by generalizing the proof in Beame et al. [BCE+98] for the case of $p = 2$. More importantly, we show the (less trivial) converse.

NS refutations from PPA_p–decision trees. The following is a query analog of Lemma 8.13.

Lemma 8.16. $\mathsf{NS}_{\mathbb{F}_p}(F) \leq O(\mathsf{PPA}_p^{\mathsf{dt}}(S(F)))$ *for any unsatisfiable k-CNF formula F.*

Proof. Suppose $F \coloneqq \bigwedge_{i\in[m]} C_i$, and let p_i be the natural polynomial encoding of C_i so that $p_i(z) = 0$ iff $C_i(z) = 1$. Fix a PPA_p–decision tree $\mathcal{T} \coloneqq (V, \{o_v, \mathcal{T}_v\}_v)$ solving $S(F)$ with cost $d \coloneqq \mathsf{PPA}_p^{\mathsf{dt}}(S(F))$. Let $|V| \equiv t \pmod p$, where $t \neq 0$. For each $v \in V$, we can define a depth-pd decision tree $\mathcal{S}_v$ such that $\mathcal{S}_v(z) = 1$ if $\deg(v) = 0$ in G_z and $\mathcal{S}_v(z) = 0$ otherwise. (First run $\mathcal{T}_v(z)$ and then $\mathcal{T}_u(z)$ for the $p-1$ potential neighbors $u \in \mathcal{T}_v(z)$.) We can then convert each $\mathcal{S}_v$ into a degree-pd $\mathbb{F}_p$-polynomial s_v in the standard way (s_v is the sum over all accepting paths of $\mathcal{S}_v$ of the product of the literals, z_i or $(1-z_i)$, recording the query outcomes on that path). By virtue of a PPA_p–decision tree, the number of $\mathcal{S}_v$ which accept z is $\equiv t \mod p$. Thus over $\mathbb{F}_p$, we have that $\sum_{v\in V} s_v(z) = t$ for all z. Moreover, we have that $p_{i_v}(z) = 0 \Rightarrow s_v(z) = 0$ where i_v is such that $o_v = C_{i_v}$; this is because s_v is only supported on inputs z for which $o_v = C_{i_v}$ is feasible (i.e., $C_{i_v}(z) = 0$ and $p_{i_v}(z) = 1$). Thus we may factor each s_v as $q_v p_{i_v}$ for some q_v. Hence we have our refutation, $\sum_{v\in V} q_v p_{i_v} = t$. Finally, for $q'_v \coloneqq t^{-1} \cdot q_v$, we get $\sum_{v\in V} q'_v p_{i_v} = 1$. □

PPA_p–decision trees from NS refutations. We now prove the converse.

Lemma 8.17. $\mathsf{PPA}_p^{\mathsf{dt}}(S(F)) \leq \mathsf{NS}_{\mathbb{F}_p}(F)$ *for any unsatisfiable k-CNF formula F.*

Proof. Suppose $F \coloneqq \bigwedge_{i\in[t]} C_i$, and let p_i be the natural polynomial encoding C_i. For $d \coloneqq \mathsf{NS}_{\mathbb{F}_p}(F)$, let $\sum_{i\in[t]} q_i p_i + 1 = 0$ be the degree-d $\mathbb{F}_p$-Nullstellensatz refutation of F.

We define a cost-d PPA_p–decision tree $\mathcal{T} \coloneqq (V, \{o_v, \mathcal{T}_v\}_v)$ solving $S(F)$. The vertices V will be grouped into $t+1$ groups $V_0, V_1, V_2, \ldots, V_t$. The group V_0 will contain only one vertex v^*, which we think of as associated with the special constant-1 term of the refutation (the label o_{v^*} is arbitrary).

For group V_i, consider expanding the polynomial $f_i \coloneqq q_i p_i$ into a sum of monomials. We call a monomial "basic" if its constant term is 1 (eg. $z_1 z_2$ is basic, whereas $2z_3 z_4$ is not). For ease of notation, let's write $f_i = \sum_{m \in \mathcal{M}} c_{i,m} \cdot m$, where $\mathcal{M}$ is the set of all basic monomials of degree at most d (note: some coefficients might be zero). Also, let $f_0 = 1$. The group V_i will contain a total of $p|\mathcal{M}|$ many vertices, in particlar, p vertices associated with each basic monomial $m \in \mathcal{M}$. Thus, we have $|V| = tp|\mathcal{M}| + 1 \not\equiv 0 \mod p$. Each $v \in V_i$ will have $o_v \coloneqq C_i$ as its associated solution.

We now describe the hyperedges of G_z. This will implicitly also specify the decision trees $\mathcal{T}_v$.

- ▷ *Out-group.* Within group V_i, for any basic monomial m, we leave out $c_{i,m} \cdot m(z)$ out of the p vertices associated with m for potential in-group hyperedges. Since $\sum_{i=0}^{m} f_i = 0$, for any basic monomial m, it holds that $\sum_i c_{i,m} = 0$. Hence, for any choice of z, globally we have left out exactly 0 (mod p) many vertices for potential in-group hyperedges. We can partition the remaining vertices corresponding to m into p-uniform hyperedges. This can be done with a canonical choice which depends only the value of z_i's that participate in m, which requires querying at most d variables.
- ▷ *In-group.* Consider group V_i. If $p_i(z) = 1$ (i.e., $C_i(z) = 0$), then we will not add any hyperedges inside V_i. If $p_i(z) = 0$, we will add many hyperedges in a way that no vertex in V_i remains isolated. First let $\rho \coloneqq z \restriction \mathrm{vars}(p_i) \in \{0, 1, *\}^n$ be the partial assignment obtained by restricting z to the variables of p_i. Consider an equivalence relation between basic monomials defined as $m \sim m'$ iff $m(\rho) = m'(\rho)$ – e.g. for $\rho = (1, *, *, \ldots, *)$, the monomials $m = z_1 z_2$ and $m' = z_2$ are equivalent. Since $p_i(\rho) = 0$, it follows that $q_i p_i(\rho)$ is an identically zero polynomial, and hence for any equivalence class $\mathcal{M}' \subseteq \mathcal{M}$, it holds that $\sum_{m \in \mathcal{M}'} c_{i,m} = 0$. Thus, for any value of z, we can partition the vertices (not covered by out-group edges) associated with $m \in \mathcal{M}'$ into p-uniform hyperedges. This can be done with a canonical choice which depends only the value of z_i's that participate in some $m \in \mathcal{M}'$. This depends only on the value of at most d many z_i's.

It is clear from the above construction that the isolated vertices in G_z are only in V_i for which $p_i(z) = 1$. Also, the edges incident to any $v \in V_i$ associated to $m \in \mathcal{M}$ can be determined by querying the variables of p_i (which defines ρ) and m. Hence the graph G_z can be described by depth-d decision trees $\mathcal{T}_v$. □

Proof of Theorem 8.12. Part 1 follows by combining Lemmas 8.16 and 8.17. Part 2 follows by combining Lemmas 8.13, 8.14 and 8.15. □

8.2.2 PLS, Resolution Width and Circuits

The characterization for communication PLS is known in literature [Raz95, Sok17]. We provide a proof for completeness and also include a characterization for query PLS.

Theorem 8.18 (PLS, Resolution Width and Circuits)**.**

1. *For any unsatisfiable CNF F :* $\mathsf{PLS}^{\mathsf{dt}}(S(F)) = \Theta($*resolution width*$(F))$*.*
2. *For any monotone function f :* $\mathsf{PLS}^{\mathsf{cc}}(\mathsf{mKW}(f)) = \Theta(\log($*monotone circuit size*$(f)))$*.*

   Moreover, for any function f : $\mathsf{PLS}^{\mathsf{cc}}(\mathsf{KW}(f)) = \Theta(\log($*circuit size*$(f)))$*.*

Proof. For Part (2.), by the characterization of circuits using rectangle dags Lemma 4.4, it suffices to show that for any search problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$, it holds that $\mathsf{PLS}^{\mathsf{cc}}(S) = \Theta(\log \mathsf{rect\text{-}dag}(S))$.

PLS–protocols from Rectangle Dags. Given a rectangle-dag Γ solving S, we show how to construct a PLS–protocol Π for S with vertex set V same as the set of nodes of Γ. For each vertex $v \in V$, we have a feasible rectangular set R_v. Protocol Π_v on inputs (x, y) works as follows:

if $(x, y) \in R_v$: Output successor $s_v(x, y)$ as the child u of v such that $(x, y) \in R_u$ and potential $\ell_v(x, y)$ as the largest distance of v from any leaf.

if $(x, y) \notin R_v$: Output successor $s_v(x, y)$ as the root and potential $\ell_v(x, y)$ as $+\infty$ (or any integer larger than depth of the rectangle dag).

It is easy to see that Π_v can be implemented as a 2-bit protocol. To verify the correctness of the PLS–protocol, observe that the only *sinks* in $G_{x,y}$ are leaves v such that $(x, y) \in R_v$ and hence $o_v \in S^{-1}(x, y)$. This shows that $\mathsf{PLS}^{\mathsf{cc}}(S) \leq \log \mathsf{rect\text{-}dag}(S) + 1$.

Rectangle Dags from PLS–protocols. Given a PLS–protocol $\Pi = (V, \{o_v, \Pi_v\}_v)$ solving S, we construct a rectangle-dag Γ as follows: We include nodes corresponding to rectangles in $\mathcal{R} = \{R_1, \ldots, R_t\}$ obtained as leaves of Π_v for each v. Note that $t \leq \exp(|V| + \max_v |\Pi_v|) = \exp(\mathsf{PLS}^{\mathsf{cc}}(\Pi))$. Consider a rectangle $R \in \mathcal{R}$ that is obtained as a leaf of Π_v for some $v \in V$. It is labeled by a successor $u \in V$ and a potential ℓ. We transition from a rectangle R by running the protocol Π_u which further partitions R into rectangles $L_1, \ldots, L_c$ where $c = \exp(|\Pi_u|)$. Note that each $L_j = R \cap R_j$ for some R_j that was obtained as a leaf of Π_u. If the potential ℓ_j of R_j is $\geq \ell$, we label the node corresponding to L_j with label o_v making it a leaf of Γ. Else we add an edge from L_j to R_j. Finally, to introduce a root node in Γ, we choose any vertex $v_0 \in V$ and starting from the root node corresponding to the rectangle $\mathcal{X} \times \mathcal{Y}$, we run the protocol Π_{v_0} which partitions the entire domain into rectangles with leaves among the rectangles $R_1, \ldots, R_t$ above. Thus the final number of nodes in Γ is at most $\exp(|V| + 2\max_v |\Pi_v|) + \exp(|\Pi_{v_0}|) \leq \exp(O(\mathsf{PLS}^{\mathsf{cc}}(\Pi)))$, or alternately $\log \mathsf{rect\text{-}dag}(\Gamma) \leq O(\mathsf{PLS}^{\mathsf{cc}}(\Pi))$.

The directed graph defined is indeed acyclic, since the edges only go from rectangles with higher potential to lower. We now verify the correctness conditions in the definition of a rectangle dag:

- *Root:* We introduced a root corresponding to $\mathcal{X} \times \mathcal{Y}$.
- *Internal Node:* Starting from the root, we run a tree-like protocol Π_{v_0} which simply partitions rectangles. For a node corresponding to $R \in \mathcal{R}$, we first run a tree-like protocol which also partitions rectangles. The only place where we *enlarge* rectangles is in going from L_j to R_j, and this is valid because $L_j \subseteq R_j$.
- *Leaves:* The leaves correspond to rectangles $L_j = R \cap R_j$ such that $\ell_j > \ell$, thus v is a sink in $G_{x,y}$ for all inputs $(x, y) \in L_j$ and hence $o_v \in S^{-1}(x, y)$.

This concludes the proof of Part (2.). For Part (1.), by the characterization of resolution refutations using conjunction dags (Lemma 4.3), it suffices to show that for search problems $S \subseteq \{0,1\}^n \times \mathcal{O}$, it holds that $\mathsf{PLS}^{\mathsf{dt}}(S) = \Theta(\log \mathsf{conj\text{-}dag}(S))$. The proof works verbatim to the communication case, by replacing PLS–protocols by PLS–decision trees and rectangle-dags by conjunction-dags. □

8.2.3 Partial Characterizations

SoPL and Comparator Circuits. One prominent circuit model that currently lacks a characterization via a $\mathsf{TFNP}^{\mathsf{cc}}$ subclass is *comparator circuits* [MS92, CFL14]. These circuits are composed only of *comparator gates* (taking two input bits and outputting them in sorted order) and input literals (only positive literals in the monotone case).

We can show an upper bound better than $\mathsf{PLS}^{\mathsf{cc}}$ for comparator circuits. Indeed, we introduced a new class SoPL (Definition 8.10) generalizing EoPL [HY17, FGMS18] precisely for this. It is easy to adapt the characterization of circuits via $\mathsf{PLS}^{\mathsf{cc}}$ (Theorem 8.18) to show that

$$\mathsf{SoPL}^{\mathsf{cc}}(\mathsf{KW}(f)) \;\leq\; O(\log(\text{comparator circuit size}(f))).$$

However, we suspect that the converse (SoPL–protocol for $\mathsf{KW}(f)$ implies a comparator circuit) is false. We also lack a proof system that is equivalent to $\mathsf{SoPL}^{\mathsf{dt}}$.

PPAD, $\mathbb{Z}$–Nullstellensatz and $\mathbb{Z}$–partition. While we lack an exact characterization of $\mathsf{PPAD}^{\mathsf{cc}}$ in terms of a computational model, we do get a one sided relationship, namely,

Theorem 8.19 (PPAD, $\mathbb{Z}$–Nullstellensatz and $\mathbb{Z}$–partition)**.**

1. *For any unsatisfiable CNF F :* $\mathsf{PPAD}^{\mathsf{dt}}(S(F)) \geq \Omega(\mathsf{NS}_{\mathbb{Z}}(F))$.
2. *For any $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$:* $\mathsf{PPAD}^{\mathsf{cc}}(S) \geq \Omega(\log \chi_{\mathbb{Z}}(S))$.

Proof. We first prove part (2). From a PPAD–protocol $\Pi \coloneqq (V, v^*, \{o_v, \Pi_v\}_v)$, we can obtain a nondeterministic protocol Γ for S, whose accepting computations output weights in $\mathbb{Z}$ (interpreted as values of the entries of an $M \in \mathcal{M}$). The protocol Γ computes as follows on input (x, y): guess a vertex $v \in V$;

if $v \neq v^*$: if v is a sink in $G_{x,y}$, accept with weight 1; if v is a source, accept with weight -1; otherwise reject (i.e., weight 0)

if $v = v^*$: if v is a source in $G_{x,y}$, reject (i.e. weight 0); if v is a sink, accept with weight 2; otherwise accept with weight 1.

In particular, Γ runs $\Pi_v(x,y)$ and then $\Pi_{p_v(x,y)}(x,y)$ and $\Pi_{s_v(x,y)}(x,y)$. The nondeterministic communication cost of Γ is thus at most 3 times that of Π. By virtue of a PPAD–protocol, it follows that Γ accepts with overall weight $\#(\text{sinks}) - \#(\text{non-distinguished sources}) = 1$ on every input (x,y), thus giving us an $\mathbb{Z}$-partition.

A similar proof in the query world (compare with Lemma 8.16) establishes Part (1). □

8.3 Separations

With the characterizations of proof systems and computational models in terms of TFNP subclasses, we can reinterpret the lifting theorems studied in this thesis as follows: For any $S \subseteq \{0,1\}^n \times \mathcal{O}$, it holds for $m \geq n^{\Delta}$ for sufficiently large constant Δ, that

$$\begin{aligned}
\mathsf{FP}^{\mathsf{cc}}(S \circ \textsc{Ind}_m^n) &= \Theta(\mathsf{FP}^{\mathsf{dt}}(S) \cdot \log n) && \text{[RM99] : Theorem 2.9} \\
\mathsf{PLS}^{\mathsf{cc}}(S \circ \textsc{Ind}_m^n) &= \Theta(\mathsf{PLS}^{\mathsf{dt}}(S) \cdot \log n) && \text{Chapter 5 : Theorem 5.1} \\
\mathsf{PPA}_p^{\mathsf{cc}}(S \circ \textsc{Ind}_m^n) &= \Theta(\mathsf{PPA}_p^{\mathsf{dt}}(S) \cdot \log n) && \text{[PR18] : Theorem 7.5}
\end{aligned}$$

Thus, we can prove separations between communication analogs of TFNP sub-classes by lifting known separations between query analogs of corresponding classes. We discuss the separations indicated in Figure 8.1.

$\mathsf{PLS}^{\mathsf{cc}} \not\subseteq \mathsf{PPA}_p^{\mathsf{cc}}$. Pitassi and Robere [PR18] showed that for any field $\mathbb{F}_p$, the function $\textsc{Gen}$ requires monotone $\mathbb{F}_p$-span programs of exponential size. On the other hand, $\textsc{Gen}$ is computable by polynomial sized monotone circuits. Thus $\mathsf{mKW}(\textsc{Gen})$ is in $\mathsf{PLS}^{\mathsf{cc}}$, but not in $\mathsf{PPA}_p^{\mathsf{cc}}$.

In hindsight, $\textsc{Gen}$ is precisely $\textsc{Horn-Sat}$, i.e. $\mathcal{C}$-$\textsc{Sat}$ for $\mathcal{C} = \textsc{Horn}$ consisting of Horn clauses.

$\mathsf{PPA}_p^{\mathsf{cc}} \not\subseteq \mathsf{PPAD}^{\mathsf{cc}}$. Pitassi and Robere [PR18] exhibit a monotone f (in hindsight, one can take $f \coloneqq 3\textsc{Lin}(\mathbb{F}_p)\text{-}\textsc{Sat}_n$) computable with a small monotone $\mathbb{F}_p$-span program (hence $\mathsf{mKW}(f) \in \mathsf{PPA}^{\mathsf{cc}}$) and such that $\mathsf{mKW}(f)$ has an exponentially large $\mathbb{R}$-*partition number*. However, we showed in Theorem 8.19 that all problems in $\mathsf{PPAD}^{\mathsf{cc}}$ have a small $\mathbb{R}$-partition number.

$\mathsf{PPAD}^{\mathsf{cc}} \not\subseteq \mathsf{PLS}^{\mathsf{cc}}$. Consider the $\mathbb{R}$-linear system $F = F_{G,\mathbb{R}}$ defined in the proof of Lemma 5.10. We now observe that $S(F_{\text{bool}})$ is in fact equivalent to (a query version of) the PPAD-complete End-of-Line problem. In the End-of-Line problem, we are given a directed graph of in/out-degree at most 1 and a distinguished source vertex v^* (in-degree 0); the goal is to find a sink or a source distinct from v^* (cf. Definition 8.6). On the other hand, in $S(F_{\text{bool}})$ we are given a boolean assignment $z \in \{0,1\}^E$, which can be interpreted as (the indicator vector of) a subset of edges defining a (spanning) subgraph G_z of G; the goal is to find a vertex $v \in V$ such that either

(1) $v = v^*$ and out-deg$(v) \neq$ in-deg$(v) + 1$ in G_z; or

(2) $v \neq v^*$ and out-deg$(v) \neq$ in-deg(v) in G_z.

The only essential difference between $S(F_{\text{bool}})$ and End-of-Line is that the graph G_z can have in/out-degree a large constant $k/2$ rather than 1. But there is a standard reduction between the two problems [Pap94]: we may locally interpret a vertex $v \in V(G_z)$ with out-deg$(v) =$ in-deg$(v) = \ell$ as ℓ distinct vertices of in/out-degree 1. This reduction also shows that the lifted problem $S(F_{\text{bool}}) \circ \textsc{Ind}_m$ for $m = n^{O(1)}$ admits a $O(\log n)$-cost PPAD-protocol, and is thus in $\mathsf{PPAD}^{\mathsf{cc}}$. By contrast, it follows similar to the proof of Lemma 5.10 that (for an appropriate choice of graph G) $\mathsf{PLS}^{\mathsf{dt}}(S(F_{\text{bool}})) \geq \Omega(n)$ and hence by the PLS–lifting theorem, $S(F_{\text{bool}}) \circ \textsc{Ind}_m$ is not in $\mathsf{PLS}^{\mathsf{cc}}$.

$\mathsf{PPADS}^{\mathsf{cc}} \not\subseteq \mathsf{PPA}_p^{\mathsf{cc}}$ via End-of-ℓ-Lines. Recall the CSP $F = F_{G,U,\mathbb{R}}$ defined in the proof of Lemma 7.2. For its booleanized version F_{bool}, it holds that $S(F_{\text{bool}})$ is infact in the query class $\mathsf{PPADS}^{\mathsf{dt}}$ (Definition 8.7). In particular, in the PPADS–decision tree, we can define the distinguished vertex v^* as being associated with any vertex from U. Similarly, the lifted problem $\widetilde{S} \coloneqq S(F_{\text{bool}}) \circ \textsc{Ind}_n^m$ for $m = n^{O(1)}$ is in the communication class $\mathsf{PPADS}^{\mathsf{cc}}$. By contrast, we proved in Lemma 7.2 that $\mathsf{PPA}_p^{\mathsf{dt}}(S(F_{\text{bool}})) \geq n^{\Omega(1)}$ and hence by the PPA_p–lifting theorem we get that $\chi_{\mathbb{F}_p}(\widetilde{S}) \geq n^{\Omega(1)}$, which implies that $\widetilde{S} \notin \mathsf{PPA}^{\mathsf{cc}}$.

$\mathsf{TFNP}^{\mathsf{cc}} \not\subseteq \mathsf{PPP}^{\mathsf{cc}}$. We claim that any problem $S \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ that lies in one of the known subclasses of $\mathsf{TFNP}^{\mathsf{cc}}$ (defined in Section 8.1) reduces efficiently to $\mathsf{mKW}(k\textsc{Cnf-Sat}_n)$ for constant k (one can even take $k = 3$ by standard reductions). We sketch the argument for $S \in \mathsf{PPP}^{\mathsf{cc}}$; after all, better reductions are known for $\mathsf{PLS}^{\mathsf{cc}}$ and $\mathsf{PPA}_p^{\mathsf{cc}}$, namely to Horn-Sat and 3Lin($\mathbb{F}_p$)-Sat (see Lemma 8.14).

Proof sketch. Let $\Pi \coloneqq (V, v^*, \{o_v, \Pi_v\}_v)$ be a PPP–protocol solving S of cost $c \coloneqq \mathsf{PPP}^{\mathsf{cc}}(S)$. We may assume wlog that all the Π_v have constant communication cost $k \leq O(1)$ by embedding the protocol trees of the Π_v as part of the implicitly described bipartite graph. In particular, we view each Π_v as a function $\mathcal{X} \times \mathcal{Y} \to \{0,1\}^k$ where the output is interpreted according to some fixed map $\{0,1\}^k \to V$. Consider a set of $n \coloneqq k|V|$ ($|V| \leq 2^c$) boolean variables $\{z_{v,i} : (v,i) \in V \times [k]\}$ with the intuitive interpretation that $z_{v,i}$ is the i-th output bit of Π_v. We may encode the correctness conditions for Π as an unsatisfiable $2k$-CNF formula F over the $z_{v,i}$ that has, for each $\{v,u\} \in \binom{V}{2}$, clauses requiring that the outputs of Π_v and Π_u (as encoded by the $z_{v,i}$) should point to distinct vertices. Finally, we note that computing the i-th output bit $(\Pi_v)_i \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ reduces to a large enough constant-size index gadget $\textsc{Ind}_{O(1)}$ (which embeds any two-party function of communication complexity $k \leq O(1)$). Therefore S naturally reduces to $S(F) \circ \textsc{Ind}^n_{O(1)}$, which by Lemma 5.9 reduces to $\mathsf{mKW}(2k\textsc{Cnf-Sat}_{O(n)})$, as desired.

Query-only separations. Beame et al. [BCE$^+$98] showed that $\mathsf{PPP}^{\mathsf{dt}} \not\subseteq \mathsf{PPADS}^{\mathsf{dt}}$ and $\mathsf{PPA}^{\mathsf{dt}} \not\subseteq \mathsf{PPP}^{\mathsf{dt}}$. The latter was generalized by Johnson [Joh11] to show that $\mathsf{PPA}_p^{\mathsf{dt}} \not\subseteq \mathsf{PPP}^{\mathsf{dt}}$. A fascinating open problem is to show $\mathsf{PLS}^{\mathsf{dt}} \not\subseteq \mathsf{PPP}^{\mathsf{dt}}$; some progress has been made in this direction [BM04].

Chapter 9

Summary and Open Problems

In this chapter, we identify some key open problems highlighted in the context of this thesis.

9.1 Lifting Theorems

More Dags. In this thesis, we proved lifting theorems for dag-like communication protocols where the feasible set corresponded to rectangles and triangles. Can our methods be extended to prove lower bounds for dags whose feasible sets are *intersections of k triangles* for $k \geq 2$? See Figure 4.2. This would imply lower bounds for proofs systems such as width-k Resolution over Cutting Planes [Kra98] and Resolution over linear equations [RT08, IS14].

Question 1. *Prove a lifting theorem for $\mathcal{F}$-dags where $\mathcal{F} := \{$intersections of k triangles$\}$.*

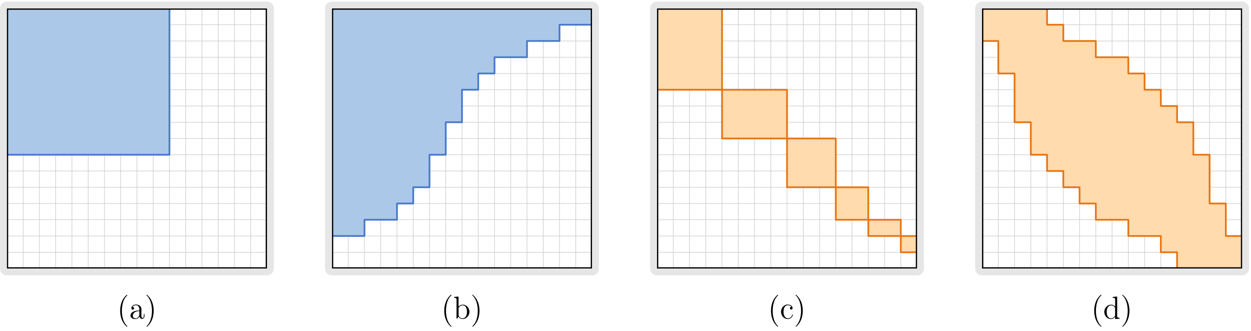


Figure 9.1: We show lifting theorems for dags whose feasible sets are (a) *rectangles* or (b) *triangles*. It remains open to prove any lower bounds for explicit mKW/CNF search problems when the feasible sets are (c) *block-diagonal*, which is a special case of (d) *intersections of 2 triangles*.

Number-on-Forehead Protocols. One of the most important open problems (e.g., [Raz16, §5]) regarding semi-algebraic proof systems that manipulate low-degree polynomials—where $\mathcal{F}$ is, say, degree-d polynomial threshold functions—is to prove lower bounds on their *dag-like* refutation length (*tree-like* lower bounds are known [BPS07, GP14]). Since degree-d polynomials can be efficiently evaluated by $(d+1)$-party number-on-forehead (NOF) protocols, one might hope to prove a dag-like NOF lifting theorem. However, we currently lack a good understanding of NOF lifting even in the tree-like case. We believe the first necessary step should be to settle the following (a two-party analogue of which was proved in [GLM^{+}16]).

Question 2. *Prove a* nondeterministic *lifting theorem for NOF protocols.*

Gadget Size. In Lemma 5.7 we showed that for a k-CNF F with ℓ clauses and resolution width w, the composed search problem $S(F) \circ \text{IND}_m^n$ reduces to $\mathsf{mKW}(f)$ for a monotone function $f : \{0,1\}^N \to \{0,1\}$ on $N \leq \ell \cdot (2m)^k$ bits for which we get an $n^{\Omega(w)}$ lower bound. Suppose for a moment that a version of Theorem 5.1, proving a $2^{\Omega(w)}$ lower bound, held for a gadget of constant size $m = O(1)$. Then we could lift any of the known CNF contradictions with parameters $k = O(1)$, $\ell = O(n)$, $w = \Omega(n)$, to obtain an explicit monotone function on $N = \Theta(n)$ variables, with essentially maximal monotone circuit complexity $2^{\Omega(N)}$, whereas the current best lower bounds for an explicit monotone functios is $2^{N^{1/3-o(1)}}$ due to Harniz and Raz [HR00].

Question 3. *Prove a lifting theorem ($\sim$ Theorems 5.1 and 6.1) with $O(1)$-sized gadgets.*

A weaker question would be to understand lifting theorem with smaller gadget size. A popular choice of gadget has been the Inner Product gadget on $O(\log n)$ bits [GLM^{+}16, CFK^{+}19].

Question 4. *Prove a lifting theorem for dag models with the $O(\log n)$-bit Inner Product gadget.*

TFNP sub-classes. There are no lower bounds known against $\mathsf{PPADS}^{\mathsf{cc}}$ or $\mathsf{PPP}^{\mathsf{cc}}$ for an explicit problem in $\mathsf{TFNP}^{\mathsf{cc}}$. For example, we don't know how to show that $\mathsf{PLS}^{\mathsf{cc}} \not\subseteq \mathsf{PPADS}^{\mathsf{cc}}$ or $\mathsf{PPA}^{\mathsf{cc}} \not\subseteq \mathsf{PPADS}^{\mathsf{cc}}$.

As described in this thesis, we have lifting theorems for FP, PLS and PPA (in fact, PPA_p). An approach to prove the above is to prove a lifting theorem for PPADS which would then allow us to lift the corresponding query separations.

Question 5. *Prove a lifting theorem for* PPADS *(or* PPP*).*

9.2 Direct Lower Bound Methods

Monotone Circuits for Perfect Matching. The Perfect-Matching function on n vertices was shown to require monotone circuits of size $n^{\Omega(\log n)}$ by Razborov [Raz85a]. It has been conjectured that the correct answer is in fact exponential (cf. [Juk12, Research Problem 9.39]). While lifting theorems have been powerful, it is unclear how to reduce a suitable composed search problem to the mKW search problem for Perfect-Matching. But perhaps, it is possible to prove the lower bound directly using the language of dag protocols and ideas inspired by our proof techniques.

Question 6. *Prove that* $\text{Perfect-Matching}_n$ *requires monotone circuits of size* $2^{n^{\Omega(1)}}$*.*

Cutting Planes refutations for Tseitin. A fascinating open problem that is still left open despite our lifting theorem for triangle-dags, is the task of proving an exponential lower bound on the Cutting Planes refutation length for Tseitin instances (cf. proof of Corollary 5.3). Interestingly, relaxing threshold-dags to triangle-dags is quite lossy here, since for any Tseitin instance F and any bipartition of the input variables, $S(F)$ does have small triangle-dags and in fact, low cost (tree-like) communication protocols! Thus, any technique to prove Cutting Planes length lower bound for Tseitin instances must work directly with threshold-dags.

Question 7. *Prove that* Tseitin_n *instances require Cutting Planes refutations of length* $2^{n^{\Omega(1)}}$*.*

9.3 Characterizations of TFNP sub-classes

Approaching from Proof Complexity. We have seen that Resolution depth/width and Nullstellensatz degree are captured by TFNP sub-classes FP/PLS and PPA_p respectively.

Question 8. *Are there* TFNP *sub-classes that are equivalent to other proof systems?*

Approaching from Monotone Complexity. We have seen that monotone formulas, circuits and span programs are captured by TFNP sub-classes FP, PLS and PPA_p respectively.

Question 9. *Are there* TFNP *sub-classes equivalent to other monotone computational models?*

Approaching from TFNP. We can ask whether there are "natural" proof systems or computational models equivalent to other well-studied TFNP sub-classes. Although, it is not to be taken for granted that such models should exist. For example, a natural property of computational models is that any partial monotone function $f : \{0,1\}^n \to \{0,1,*\}$ can be completed into a total monotone function $\widetilde{f} : \{0,1\}^n \to \{0,1\}$ without increasing the computational cost of the model, since a computational model can be evaluated on inputs in $f^{-1}(*)$ to produce *some* output in $\{0,1\}$. But it is not clear how to use a protocol for $\mathsf{mKW}(f)$ for a partial f to extend it to a total $\widetilde{f}$, since the protocol is undefined if either player gets an input in $f^{-1}(*)$.

Question 10. *What are (if any) proof systems / monotone computational models equivalent to* EoML*,* SoML*,* PPAD*,* PPADS *and* PPP*?*

Bibliography

[AB87] Noga Alon and Ravi Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987. doi:10.1007/BF02579196.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

[ABK17] Jayadev Acharya, Arnab Bhattacharyya, and Pritish Kamath. Improved bounds for universal one-bit compressive sensing. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 2353–2357, 2017. doi:10.1109/ISIT.2017.8006950.

[ABRW04] Michael Alekhnovich, Eli Ben-Sasson, Alexander Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004. doi:10.1137/S0097539701389944.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1):2:1–2:54, 2009. doi:10.1145/1490270.1490272.

[BCE$^+$98] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 337–347, 1986. doi:10.1109/SFCS.1986.15.

[BGH$^+$16] Mohammad Bavarian, Badih Ghazi, Elad Haramaty, Pritish Kamath, Ronald L. Rivest, and Madhu Sudan. The optimality of correlated sampling. *Manuscript*, 2016. arXiv:1612.01041.

[BGIP01] Sam Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, 2001. doi:10.1006/jcss.2000.1726.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–442, 1975. doi:10.1137/0204037.

[BHP10] Paul Beame, Trinh Huynh, and Toniann Pitassi. Hardness amplification in proof complexity. In *Proceedings of the 42nd Symposium on Theory of Computing (STOC)*, pages 87–96. ACM, 2010. doi:10.1145/1806689.1806703.

[Bla38] Archie Blake. *Canonical Expressions in Boolean Algebra.* University of Chicago Press, 1938. doi:10.2307/2267634.

[BM04] Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

[BP01] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. In *Current Trends in Theoretical Computer Science: Entering the 21st Century*, pages 42–70. World Scientific, 2001. doi:10.1142/9789812810403_0001.

[BPR97] Maria Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997. doi:10.2307/2275569.

[BPS07] Paul Beame, Toniann Pitassi, and Nathan Segerlind. Lower bounds for Lovász–Schrijver systems and beyond follow from multiparty communication complexity. *SIAM Journal on Computing*, 37(3):845–869, 2007. doi:10.1137/060654645.

[BR98] Paul Beame and Søren Riis. More on the relative strength of counting principles. In *Proceedings of the DIMACS Workshop on Proof Complexity and Feasible Arithmetics*, volume 39, pages 13–35, 1998.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

[CCT87] William Cook, Collette Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.

[CFK$^+$19] Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-To-Communication Lifting for BPP Using Inner Product. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.35.

[CFL14] Stephen Cook, Yuval Filmus, and Dai Tri Man Lê. The complexity of the comparator circuit value problem. *ACM Transactions on Computation Theory*, 6(4):15:1–15:44, 2014. doi:10.1145/2635822.

[Chu36] Alonzo Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936. doi:10.2307/2269326.

[Chv73] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Math.*, 4(1):305–307, 1973.

[CKLM17] Arkadev Chattopadhyay, Michal Koucky, Bruno Loff, and Sagnik Mukhopadhyay. Composition and simulation theorems via pseudo-random properties. *Electronic Colloquium on Computational Complexity (ECCC)*, (14), 2017. URL: https://eccc.weizmann.ac.il/report/2017/014/.

[CLRS13] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate constraint satisfaction requires large LP relaxations. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 350–359, 2013. doi:10.1109/FOCS.2013.45.

[Coo71] 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. doi:10.1145/800157.805047.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979. doi:10.2307/2273702.

[DM18] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960. doi:10.1145/321033.321034.

[DP11] Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 790–804, 2011. doi:10.1137/1.9781611973082.62.

[dRMN$^+$19] Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. *Manuscript*, 2019.

[dRNV16] Susanna de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity).

In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 295–304. IEEE, 2016. doi:10.1109/FOCS.2016.40.

[FGMS18] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. End of potential line. *arXiv*, 1804.03450, 2018. URL: http://arxiv.org/abs/1804.03450.

[FPPR17] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random CNFs are hard for cutting planes. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, 2017. doi:10.2307/2275569.

[Gál01] Anna Gál. A characterization of span program size and improved lower bounds for monotone span programs. *Computational Complexity*, 10(4):277–296, 2001. doi:10.1007/s000370100001.

[GGKS18] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 902–911, 2018. doi:10.1145/3188745.3188838.

[GHKS17] Badih Ghazi, Elad Haramaty, Pritish Kamath, and Madhu Sudan. Compression in a distributed setting. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 19:1–19:22, 2017. doi:10.4230/LIPIcs.ITCS.2017.19.

[GKPW19] Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for P NP. *Computational Complexity*, 28(1):113–144, 2019. doi:10.1007/s00037-018-0175-5.

[GKR18] Badih Ghazi, Pritish Kamath, and Prasad Raghavendra. Dimension reduction for polynomials over gaussian space and applications. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 28:1–28:37, 2018.

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations in Theoretical Computer Science*

Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA, pages 38:1–38:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

[GKS16a] Badih Ghazi, Pritish Kamath, and Madhu Sudan. Communication complexity of permutation-invariant functions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1902–1921, 2016.

[GKS16b] Badih Ghazi, Pritish Kamath, and Madhu Sudan. Decidability of non-interactive simulation of joint distributions. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 545–554, 2016.

[GKSZ19] Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis. On the complexity of modulo-q arguments. *Manuscript*, 2019.

[GLM$^+$16] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016. doi:10.1137/15M103145X.

[Gom63] Ralph E. Gomory. An algorithm for integer solutions of linear programs. *Recent Advances in Mathematical Programming*, pages 269–302, 1963.

[Göö15] Mika Göös. Lower bounds for clique vs. independent set. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1066–1076, 2015. URL: https://doi.org/10.1109/FOCS.2015.69, doi:10.1109/FOCS.2015.69.

[GP14] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Proceedings of the 46th Symposium on Theory of Computing (STOC)*, pages 847–856. ACM, 2014. doi:10.1145/2591796.2591838.

[GP18a] Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *J. Comput. Syst. Sci.*, 94:167–192, 2018. doi:10.1016/j.jcss.2017.12.003.

[GP18b] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*, pages 1077–1088. IEEE, 2015. doi:10.1109/FOCS.2015.70.

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 132–143, 2017. doi:10.1109/FOCS.2017.21.

[Hås86] 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. doi:10.1145/12130.12132.

[HC99] Armin Haken and Stephen Cook. An exponential lower bound for the size of monotone real circuits. *Journal of Computer and System Sciences*, 58(2):326–335, 1999. doi:10.1006/jcss.1998.1617.

[HG18] Alexandros Hollender and Paul Goldberg. The complexity of multi-source variants of the End-of-Line problem, and the concise mutilated chessboard. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2018. URL: https://eccc.weizmann.ac.il/report/2018/120/.

[HKV15] Bernhard Haeupler, Pritish Kamath, and Ameya Velingker. Communication with partial noiseless feedback. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 881–897, 2015.

[HP17] Pavel Hrubes and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 121–131, 2017. doi:10.1109/FOCS.2017.20.

[HP18] Pavel Hrubeš and Pavel Pudlák. A note on monotone real circuits. *Information Processing Letters*, 131:15 – 19, 2018. doi:10.1016/j.ipl.2017.11.002.

[HR00] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In *Proceedings of the 32nd Symposium on Theory of Computing (STOC)*, pages 378–387. ACM, 2000. doi:10.1145/335305.335349.

[HRST17] Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *J. ACM*, 64(5):35:1–35:27, 2017. doi:10.1145/3095799.

[HY17] Pavel Hub'avcek and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1352–1371, 2017. doi:10.1137/1.9781611974782.88.

[IS14] Dmitry Itsykson and Dmitry Sokolov. Lower bounds for splittings by linear combinations. In *Proceedings of the 39th Mathematical Foundations of Computer Science (MFCS)*, pages 372–383. Springer, 2014. doi:10.1007/978-3-662-44465-8_32.

[Jaf06] A.M. Jaffe. The millennium grand challenge in mathematics. 53:652–660, 06 2006.

[Joh11] Alan S. Johnson. Reductions and propositional proofs for total NP search problems. *UC San Diego Electronic Theses and Dissertations*, 2011. URL: https://escholarship.org/uc/item/89r774x7.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

[Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972. URL: http://www.cs.berkeley.edu/%7Eluca/cs172/karp.pdf.

[KMR17] 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. doi:10.1145/3055399.3055438.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62(2):457–486, 1997. doi:10.2307/2275541.

[Kra98] Jan Krajíček. Discretely ordered modules as a first-order extension of the cutting planes proof system. *Journal of Symbolic Logic*, 63(4):1582–1596, 1998. doi:10.2307/2586668.

[Kra19] Jan Krajícek. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. doi:10.1017/9781108242066.

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995. doi:10.1007/BF01206317.

[KW88] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the 20th Symposium on Theory of Computing (STOC)*, pages 539–550. ACM, 1988. doi:10.1145/62212.62265.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 102–111, 1993. doi:10.1109/SCT.1993.336536.

[LMV17] James R. Lee, Raghu Meka, and Thomas Vidick. Personal communication. 2017.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, 1995. doi:10.1137/S0895480192233867.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *Proceedings of the 2nd Conference on Fundamentals of Computation Theory (FCT)*, pages 565–574, 1979.

[LRS15] James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 567–576, 2015. doi:10.1145/2746539.2746599.

[LS09] Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–398, 2009. doi:10.1561/0400000040.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.

[MS92] Ernst Mayr and Ashok Subramanian. The complexity of circuit value and network stability. *Journal of Computer and System Sciences*, 44(2):302–323, 1992. doi:10.1016/0022-0000(92)90024-D.

[Mul87] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987. doi:10.1007/BF02579205.

[MW18] Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901, 2018. doi:10.1145/3188745.3188910.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

[PR18] Toniann Pitassi and Robert Robere. Lifting Nullstellensatz to monotone span programs over any field. In *Proceedings of the 50th Symposium on Theory of Computing (STOC)*, pages 1207–1219. ACM, 2018. doi:10.1145/3188745.3188914.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997. doi:10.2307/2275583.

[Pud00] Pavel Pudlák. Proofs as games. *The American Mathematical Monthly*, 107(6):541–550, 2000. doi:10.2307/2589349.

[Pud10] Pavel Pudlák. On extracting computations from propositional proofs (a survey). In *Proceedings of the 30th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 30–41. Schloss Dagstuhl, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.30.

[Raz85a] Alexander Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985. doi:10.1007/BF01157687.

[Raz85b] Alexander Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk USSR*, 285:798–801, 1985.

[Raz89] Alexander A. Razborov. On the method of approximations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 167–176, 1989. URL: https://doi.org/10.1145/73007.73023, doi:10.1145/73007.73023.

[Raz95] Alexander Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya of the RAN*, pages 201–224, 1995.

[Raz16] Alexander Razborov. Proof complexity and beyond. *SIGACT News*, 47(2):66–86, 2016. doi:10.1145/2951860.2951875.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. doi:10.1007/s004930050062.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253.

[Ros97] Arnold Rosenbloom. Monotone real circuits are more powerful than monotone boolean circuits. *Information Processing Letters*, 61(3):161 – 164, 1997. doi:https://doi.org/10.1016/S0020-0190(97)00007-0.

[RS10] Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of ac^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010. doi:10.1137/080744037.

[RT08] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Annals of Pure and Applied Logic*, 155(3):194–224, 2008. doi:10.1016/j.apal.2008.04.001.

[RW92] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992. doi:10.1145/146637.146684.

[RY17] Anup Rao and Amir Yehudayoff. *Communication Complexity.* In preparation, 2017. URL: https://homes.cs.washington.edu/~anuprao/pubs/book.pdf.

[She11] Alexander Sherstov. The pattern matrix method. *SIAM Journal on Computing*, 40(6):1969–2000, 2011. doi:10.1137/080733644.

[SKLS18] Ankit Shah, Pritish Kamath, Shen Li, and Julie A. Shah. Bayesian inference of temporal task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 3808–3817, 2018.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987. doi:10.1145/28395.28404.

[Sok17] Dmitry Sokolov. Dag-like communication and its applications. In *Proceedings of the 12th Computer Science Symposium in Russia (CSR)*, pages 294–307. Springer, 2017. doi:10.1007/978-3-319-58747-9_26.

[SZ09] Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Information and Computation*, 9(5–6):444–460, 2009.

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. Ppp-completeness with connections to cryptography. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 148–158, 2018. doi:10.1109/FOCS.2018.00023.

[Tar88] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988. doi:10.1007/BF02122563.

[Tur37] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1 1937. doi:10.1112/plms/s2-42.1.230.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

[Ver99] Nikolai K. Vereshchagin. Relativizability in complexity theory. In *Provability, Complexity, Grammars*, volume 192 of *AMS Translations, Series 2*, pages 87–172. American Mathematical Society, 1999.

[Wig19] Avi Wigderson. *Mathematics and Computation.* Princeton University Press, 2019. URL: https://www.math.ias.edu/files/mathandcomp.pdf.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 115–125, 2011. doi:10.1109/CCC.2011.36.

[Wu17] Xinyu Wu. The uniform marginals lemma in [GPW17]. *Manuscript*, 2017. URL: https://www.andrew.cmu.edu/user/xinyuw1/papers/uniform-marginals-lemma.pdf.

[WYY17] Xiaodi Wu, Penghui Yao, and Henry Yuen. Raz-McKenzie simulation with the inner product gadget. *Electronic Colloquium on Computational Complexity (ECCC)*, (10), 2017. URL: https://eccc.weizmann.ac.il/report/2017/010/.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979. doi:10.1145/800135.804414.