UNIVERSITY OF CALIFORNIA SAN DIEGO

Upper and Lower bounds for Polynomial Calculus with extension variables

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Communication Theory and Systems)

by

Venkata Sai Sasank Mouli Gali

Committee in charge:

Professor Russell Impagliazzo, Chair
Professor Young-Han Kim, Co-Chair
Professor Samuel Buss
Professor Alon Orlitsky

2022

The dissertation of Venkata Sai Sasank Mouli Gali is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my parents.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Many people helped and motivated me over the years which led to my pursuing a PhD degree in theoretical computer science. I thank them here in chronological order.

I am and always will be indebted to my parents. They motivated me towards academics from a young age, and always took pride in my academic achievements. Thank you for always being there for me.

I am especially thankful to my friend Amit Munje, with whom I had invigorating discussions on math and theoretical computer science during my undergrad. I could say that he is directly responsible for my pursuit of graduate studies in this area.

My advisors Russell Impagliazzo and Toniann Pitassi are amongst the smartest yet most gentle people I have ever met. Their guidance was invaluable and taught me a lot about how to think and approach research problems. Thank you for supporting me through thick and thin.

Last but not least, I have to offer much love to Jing Liang, my partner for a couple of years during my graduate studies. The time we spent together in my apartment will be forever etched into my memory.

Many other people supported me in so many ways. Thanks to all of them, including but not limited to: Vaishakh Ravi, Shouvik Ganguly, Anant Dhayal, Jess Sorrell, Rex Lei, Sam McGuire, Sam Buss, Marco Carmosino, Jiawei Gao, Anamika Agrawal and Anchal Gupta.

Chapters 1,2 contain material from "The surprising power of constant depth Algebraic Proofs" by Russell Impagliazzo, Sasank Mouli and Toniann Pitassi, published at the Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, 2020.

Chapters 1,3 contain material from "Lower bounds for Polynomial Calculus with extension variables over finite fields" by Russell Impagliazzo, Sasank Mouli and Toniann Pitassi which is currently published on the Electronic Colloquium on Computational Complexity and being prepared for a formal publication.

VITA

2016	B. Tech. in Electrical Engineering with a second major in Computer Science, Indian Institute of Technology, Kanpur, India
2018	M. S. in Electrical Engineering (Communication Theory and Systems), University of California San Diego
2022	Ph. D. in Electrical Engineering (Communication Theory and Systems), University of California San Diego

PUBLICATIONS

Impagliazzo, Russell, Sasank Mouli, and Toniann Pitassi, "The surprising power of constant depth Algebraic Proofs", *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, 2020.

ABSTRACT OF THE DISSERTATION

Upper and Lower bounds for Polynomial Calculus with extension variables

by

Venkata Sai Sasank Mouli Gali

Doctor of Philosophy in Electrical Engineering (Communication Theory and Systems)

University of California San Diego, 2022

Professor Russell Impagliazzo, Chair
Professor Young-Han Kim, Co-Chair

A major open problem in proof complexity is to prove superpolynomial lower bounds for $\mathsf{AC}^0[p]$-Frege proofs. This system is the analog of $\mathsf{AC}^0[p]$, the class of bounded depth circuits with prime modular counting gates. Despite strong lower bounds for this class dating back thirty years ([Raz87, Smo87]), there are no significant lower bounds for $\mathsf{AC}^0[p]$-Frege. Significant and extensive *degree* lower bounds have been obtained for a variety of subsystems of $\mathsf{AC}^0[p]$-Frege, including Nullstellensatz ([BIK$^+$94]), Polynomial Calculus ([CEI96]), and SOS ([GV01]). However to date there has been no progress on $\mathsf{AC}^0[p]$-Frege lower bounds.

In the first part of this thesis, we study constant-depth extensions of the Polynomial

Calculus [GH03]. We show that these extensions are much more powerful than was previously known. Our main result is that small depth (≤ 43) Polynomial Calculus (over a sufficiently large field) can polynomially effectively simulate all of the well-studied semialgebraic proof systems: Cutting Planes, Sherali-Adams, Sum-of-Squares (SOS), and Positivstellensatz Calculus (Dynamic SOS). Additionally, they can also quasi-polynomially effectively simulate $\mathsf{AC}^0[q]$-Frege for *any* prime q independent of the characteristic of the underlying field. They can also effectively simulate TC^0-Frege if the depth is allowed to grow proportionally. Thus, proving strong lower bounds for constant-depth extensions of Polynomial Calculus would not only give lower bounds for $\mathsf{AC}^0[p]$-Frege, but also for systems as strong as TC^0-Frege.

In the second part of this thesis, we deal with the flip side of proving new lower bounds for these systems. In this direction, we extend the recent lower bounds of Sokolov [Sok20] for Polynomial Calculus over $\{\pm 1\}$ variables to show for every prime $p > 0$, every $n > 0$ and $\kappa = O(\log n)$ the existence of an unsatisfiable system of polynomial equations over $O(n \log n)$ variables of degree $O(\log n)$ such that any Polynomial Calculus refutation over $\mathbb{F}_p$ with M extension variables, each depending on at most κ original variables requires size $\exp\left(\Omega(n^2/(\kappa^2 2^\kappa(M + n\log(n))))\right)$.

Chapter 1

Introduction

Propositional Proof Complexity is the study of the complexity of proving Boolean tautological statements in systems of formal reasoning which deal with propositional formulae. A more formal definition is given below.

Definition 1 (Boolean formula)**.** *A Boolean formula is a directed acyclic graph, which is a binary tree, with each node labeled with a gate ($\wedge$ or $\vee$), and each leaf node labeled with a literal (x or $\neg x$ for a variable x). The size of the formula is the number of leaves, and the depth of the formula is the longest path from the root to a leaf.*

Definition 2 ($TAUT$)**.** *Let $TAUT$ be the set of all tautological Boolean formulae, i.e. those that always evaluate to TRUE.*

Definition 3 (Propositional Proof System)**.** *A Propositional Proof System is a polynomial time verifier V such that for every formula φ in $TAUT$, there exists a proof Π such that V accepts on Π.*

The above condition is sometimes referred to as *Completeness*. Another condition called *Soundness* is also stipulated of Propositional Proof Systems, which requires that only tautologies have valid proofs in the system. The study of Propositional Proof Systems was first initiated by the

work of Cook and Reckhow [CR79], with the ultimate goal of obtaining a proof of $\mathsf{coNP} \not\subseteq \mathsf{NP}$ by proving a lower bound for a tautology against increasingly strong proof systems. This is analogous to the goal of circuit complexity, which aims to prove that NP is different from P by proving lower bounds against increasingly strong circuit classes. This parallel with circuit complexity also extends to known results: in both these areas lower bounds are known only for weak, computationally restricted classes. In proof complexity, this means that each line of the proof in question is restricted computationally, i.e. can be computed by a small size circuit of limited complexity.

One of the weakest, and the most well studied Propositional Proof System is Resolution, which deals with proof lines which are clauses i.e. ORs ($\vee$) of literals. This proof system was invented to try and capture the performance of a popular class of satisfiability solvers known as David-Putnam-Logemann-Loveland (DPLL) algorithms [Mar62]. In Resolution, two clauses can be combined using the *Resolution rule* to derive a new clause as follows. Given two clauses $C \vee x$ and $D \vee \neg x$, we derive $C \vee D$. This rule can be easily shown to be complete and sound.

Resolution falls at the bottom of a hierarchy of proof systems known as the *Frege* systems. In the Frege system, each line is represented by a Boolean formula, and there are natural inference rules which allow deriving more complex lines. Similar to the situation with circuit complexity, here lower bounds are known only for Frege systems whose lines are *bounded-depth* formulae. This system is known as AC^0-Frege.

1.1 Algebraic Proof Systems

Although lower bounds are known for Resolution [Hak85], [BSW99], and bounded-depth Frege [Ajt94], [BIK$^+$92], the crucial difference between circuit and proof complexity arrives at the next step of obtaining lower bounds for bounded-depth with modular counting gates of a particular prime modulus p, known as $\mathsf{AC}^0[p]$. Lower bounds are known for this system in circuit

complexity through the work of Razborov and Smolensky [Raz87, Smo87], but no known lower bounds exist for the corresponding proof system $\mathsf{AC}^0[p]$-Frege. Since the latter lower bounds were obtained through connections to algebraic objects, this suggested the study of *Algebraic Proof Systems*.

Typically, algebraic proof systems work by encoding a tautological formula φ as a system of unsatisfiable polynomial equations $P_1 = 0, \ldots, P_m = 0$. *Nullstellensatz* was one of the first algebraic proof systems to be introduced. The completeness of this system is guaranteed by *Hilbert's Nullstellensatz*, which provides a set of polynomials $Q_1 = 0, \ldots Q_m = 0$ for the former system, such that $\sum_i P_i Q_i = 1$. The latter set of polynomials serves as a *certificate of unsatisfiability* for the former. The complexity of a Nullstellensatz proof of $P_1 = 0, \ldots, P_m = 0$ is the size (i.e. number of monomials) of the certificate of their unsatisfiability.

Nullstellensatz was generalized by [CEI96] into the *Polynomial Calculus* proof system, where instead of the certificate of unsatisfiability being provided all at once, is derived dynamically by searching the degree d *pseudo ideal* of $P_1 = 0, \ldots, P_m = 0$. This is done by the following two rules. From two polynomials P and Q, we can derive their linear combination $\alpha P + \beta Q$, where α, β are constants from the underlying field. Also, from a polynomial P and a variable x, we can derive $xP = 0$, provided that the degree of xP is bounded by d. The goal is to derive a contradictory line $1 = 0$.

1.2 Extensions of the Polynomial Calculus

Once again, we return to our original goal of proving lower bounds for the system $\mathsf{AC}^0[p]$-Frege. Given that there are lower bounds for the Polynomial Calculus [Raz98a, BGIP01, AR01, GL10, MN15], progress towards the frontier of $\mathsf{AC}^0[p]$-Frege is still stalled for various reasons. Since lower bounds for polynomial calculus itself do not imply lower bounds for $\mathsf{AC}^0[p]$-Frege systems, various researchers have suggested ways to strengthen PC to create algebraic systems

which do p-simulate $\mathsf{AC}^0[p]$-Frege ([Pit96, GH03, BIK+97]). Unfortunately, it is not clear how to extend lower bound techniques for PC to these systems. As an illustration of how small extensions can increase the power of these proof systems, consider polynomial calculus where we allow changes of bases. Many strong lower bounds are known for the size of PC proofs for tautologies like the Pigeonhole Principle [Raz98b], [IPS99] and Tseitin tautologies [BGIP01]. All of the above lower bounds use a degree-size connection, which roughly states that a linear lower bound on the degree of any refutation translates to an exponential lower bound on its size. But this connection is highly basis dependent. The connection only holds true over the $\{0,1\}$ basis, and even allowing a change to the $\{-1,1\}$ basis immediately gives a polynomial sized proof for the *mod* 2 Tseitin tautologies. Grigoriev and Hirsch [GH03] noted the above and in addition showed that allowing for introduction of new variables which are linear transformations of the original variables gives a short proof of the Pigeonhole principle as well. They also generalized the notion of a linear transformation by considering transformations obtained by applying constant depth arithmetic circuits and arithmetic formulas to the original variables. The resulting systems turn out to be quite powerful, and it is shown in [GH03] that the latter simulates Frege systems, and the former simulates depth d $\mathsf{AC}^0[p]$-Frege proofs by using arithmetic circuits of depth $d' = \Theta(d)$. Raz and Tzameret [RT08b] defined a proof system along similar lines where the transformations are restricted such that each line of the proof is a multilinear formula in the original variables. It was shown that even under these restrictions, linear transformations allow small proofs of the functional Pigeonhole principle and Tseitin tautologies. They also showed in [RT08a] that Polynomial Calculus with added linear transformations simulates the system $R(CP^*)$ of Krajicek [Kra98], which is stronger than Cutting Planes with bounded coefficients.

1.3 New lower bound techniques

On the flip side, the lower bound techniques used for the above lower bounds on Polynomial Calculus are based on random restrictions, and it is well known that modular counting gates are immune to such techniques. We also need lower bound techniques that are not just random-restriction based. Sokolov [Sok20] obtained lower bounds for the system PC over $\mathbb{F}_3$ where extension variables of the form $z_i = 2x_i - 1$ are allowed to be introduced (hence making them take values in the set $\{+1, -1\}$), using techniques which are not random restriction based.

1.4 Contributions of this work

The contributions of this work are twofold: improved upper bounds and lower bounds for Polynomial Calculus with extension variables.

1.4.1 Upper bounds

On the one hand, we show that these extensions to PC are even more powerful than previously known. Over a sufficiently large field of characteristic p, the same extensions that allow PC to simulate depth d $\mathsf{AC}^0[p]$ proofs also allows it to simulate much stronger proof systems. So to prove a lower bound on $\mathsf{AC}^0[p]$ proofs via such systems would seem to require proving lower bounds for systems as strong as TC^0-Frege.

More precisely, consider the following additions to PC. In an additive extension, we introduce a new variable y and a new defining equation $y = \sum a_i x_i + b$ where $a_i, b \in \mathbb{F}$. In a multiplicative extension, we introduce a new variable y and a new defining equation $y = b\prod(x_i)^{e_i}$. Depth-d-PC allows the usual (syntactic) reasoning of Polynomial calculus using these extension variables (i.e. multiplying a line by the variable y is allowed), with each line having up to $d-2$ layers of additive and multiplicative extensions, the layers alternating between them. (The new

variables in a depth d-PC proof are equivalent to depth $d-2$ algebraic circuits, and polynomials in terms of these variables are depth d algebraic circuits.)

We improve the results of Raz and Tzameret [RT08a] to show that Polynomial Calculus with linear transformations can simulate *semantic* Cutting Planes with small coefficients.

Theorem 1. *(Informal)*

Polynomial Calculus over $\mathbb{Q}$ where new variables defined by linear transformations are allowed to be introduced can p-simulate semantic Cutting Planes with polynomially bounded coefficients

We also show that their simulation of syntactic Cutting Planes can be carried out over a large enough field extension $\mathbb{F}_{p^m}$ (See Section 2.5.1).

We improve the results of Grigoriev and Hirsch in the constant depth case in two ways. We show that $\mathsf{AC}^0[p]$-Frege can be simulated with a fixed constant depth, but with a quasipolynomial blowup. Significantly, this simulation also simulates modular gates of different characteristic than that of the field we are working over.

Theorem 2. *(Informal)*

There is a fixed constant d such that Polynomial Calculus over a quasipolynomial sized extension field $\mathbb{F}_{p^m}$, where new variables defined by depth d arithmetic formulas on the original variables are allowed to be introduced, quasipolynomially simulates $\mathsf{AC}^0[q]$-Frege, for any prime q.

Buss *et al.* [BKZ15] showed that an $\mathsf{AC}^0[p]$-Frege proof of depth d can be collapsed to a depth 3 $\mathsf{AC}^0[p]$-Frege proof with a quasipolynomial blowup. In conjunction with [GH03], this implies the above theorem for the case of $q = p$. Thus, apart from being more general, our result also provides an alternative and perhaps simpler proof of the case of $q = p$.

We also show that allowing for arbitrarily large but constant depth transformations enables the simulation of TC^0-Frege.

Theorem 3. *(Informal)*

A TC_0-Frege proof of depth d can be p-simulated by a Polynomial Calculus proof which allows

new variables to be introduced which are defined by depth $O(d)$ arithmetic formulas on original variables

Finally, we remove the restriction of polynomially bounded coefficients from the result of [RT08a] and show how to perform arithmetic with large coefficients, and as a result simulate Cutting Planes with unbounded coefficients and Sum of Squares.

Theorem 4. *(Informal)*
There is a fixed constant d such that Polynomial Calculus over a large enough extension field $\mathbb{F}_{p^m}$, where new variables defined by depth d arithmetic formulas on the original variables are allowed to be introduced, p-simulates Cutting Planes and Positivestellensatz Calculus.

Clote and Kranakis [CK13] mention a proof, due to Krajíček, of Cutting Planes being simulated by the bounded-depth threshold logic system PTK of Buss and Clote [BC96]. Since we simulate a modified version of PTK to show Theorem 2, it already follows that our system simulates Cutting Planes. However, the above proof by Krajicek is non-explicit and does not provide a value of the depth at which the simulation happens. Determining this value is posed as an open problem in [CK13]. Theorem 4 provides an upper bound of $d \leq 10$ through an explicit simulation.

This work is based on the paper "Impagliazzo, Russell, Sasank Mouli, and Toniann Pitassi. "The surprising power of constant depth Algebraic Proofs." Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. 2020." and is expounded on in Chapter 2.

1.4.2 Lower bounds

On the other hand, in the case of lower bounds, we generalize the methods of Sokolov to show lower bounds for PC with up to $N^{2-\varepsilon}$ extension variables which can depend on up to $\kappa = O(\log N)$ original variables (where N is the number of variables in the tautology) (We

call these κ-local extension variables). [Ale21] obtained stronger lower bounds for Polynomial Calculus with extension variables over the reals, but since we work over finite fields our results are incomparable. Also, their tautology is a variant of subset sum with large coefficients, which cannot be defined well over finite fields.

Theorem 5 (high-end)**.** *There is a family of CNF tautologies $\psi_{N,\kappa,M}$ on N variables with $poly(N)$ clauses of width $O(logN)$ so that for any $M = Npolylog(N)$ and $\kappa \geq 1$, and prime p, there is a function $S(N) \in 2^{\Omega(N/polylog(N))}$ so that any* PC *refutation of $\psi_{N,\kappa,M}$ together with any M κ-local extensions over $\mathbb{F}_p$ requires size $S(N)$.*

Theorem 6 (low-end)**.** *For the same family of tautologies above, for any prime p, there are $0 < \alpha, \beta, \gamma < 1$, with $\gamma < 1 - \alpha - \beta$ so that, for $M = N^{1+\alpha}, \kappa = \beta logN$, and $S = exp(N^{\gamma})$, any* PC *refutation of Φ_N together with any M κ-local extensions over $\mathbb{F}_p$ requires size $S(N)$.*

This work is based on the paper "Impagliazzo, Russell, Sasank Mouli, and Toniann Pitassi. Lower bounds for Polynomial Calculus with extension variables over finite fields." and is discussed in Chapter 3.

Chapter 1 contains material from "The surprising power of constant depth Algebraic Proofs." by Russell Impagliazzo, Sasank Mouli, and Toniann Pitassi.which is published at the Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. 2020. and "Lower bounds for Polynomial Calculus with extension variables over finite fields." by Russell Impagliazzo, Sasank Mouli, and Toniann Pitassi. which is currently published on the Electronic Colloquium on Computational Complexity.

Chapter 2

The Surprising Power of Constant Depth Algebraic Proofs

2.1 Organization of this chapter

This chapter is organized as follows. In section 2.2.1, we discuss some basic definitions. In section 2.2.3, we formalize the notion of transformations. In section 2.3, we formally state all of our results. In section 2.4.1, we sketch the simulation of syntactic Cutting Planes with bounded coefficients from [RT08a], since it is essential for a significant part of the subsequent discussion. In section 2.4.2, we extend the simulation to the semantic case. In section 2.5.1, we prove an analog of the results in section 2.4.1 over a large enough finite field extension. In sections 2.5.2, 2.5.2, 2.5.3, we use techniques from this analog to prove Theorems 2 and 3. Finally in section 2.5.4, we prove Theorem 4. Technical details of simulations from each of the above sections are contained in the Appendix.

2.2 Preliminaries and Generalizations of Polynomial Calculus

2.2.1 Preliminaries

Notation

Integers are represented by letters a, b, c. For an integer a, let $a^+ = a$ if $a > 0$ and 0 otherwise. Define $|a|$ to be the length in binary of a. Sets of integers are represented by letters A, B, C. Indices to sets are represented by letters i, j, k, ℓ.

Variables are represented by x, y, z, w where x usually represents the original variables and the others represent the extension variables. Monomials are represented by upper case letters X, Y, Z. Polynomials are represented by P, Q, R. Boolean formulae are represented by φ.

We treat all the above as one dimensional objects. Multidimensional objects, or vectors, are represented in boldface. Constant vectors are represented by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$. Vectors whose components may be variables or polynomials are represented by $\mathbf{y}$, $\mathbf{z}$, $\mathbf{w}$.

Calligraphic letters $\mathcal{R}$, $\mathcal{S}$ are used for special expressions which are contextual.

Definition 4. *Straight Line Program (SLP)*

A SLP S over variables $\{x_1 \cdots x_n\}$ *and a field* $\mathbb{F}$ *is a sequence of computations* $(y_1 \cdots y_k)$ *such that each* y_j *is equal to one of the following, where* $C_j \subseteq \{1 \cdots j-1\}$

x_i *for some* $i \in \{1 \cdots n\}$

$\sum_{\ell \in C_j} \alpha_\ell y_\ell$ *for some constants* $\alpha_\ell \in \mathbb{F}$

$\prod_{\ell \in C_j} y_\ell$

We view a SLP as a directed acyclic graph where internal nodes are labelled with either Product or Plus gates and the leaf nodes are labelled with a variable x_i*. The size of a SLP is*

therefore the number of nodes in the corresponding directed acyclic graph, and the depth is the maximum number of nodes on a root to leaf path in the directed acyclic graph.

2.2.2 Propositional proof systems

Definition 5. *Cook-Rechow proof system*

For a language $L \subseteq \{0,1\}^*$, a Cook-Rechow proof system is a polynomial time deterministic verifier V such that

If $x \in L$, there exists a proof π such that $V(x,\pi)$ accepts.

If $x \notin L$, for all proofs π, $V(x,\pi)$ rejects.

Definition 6. *p-simulation*

For two proof systems V_1 and V_2 defined over the same language L, V_2 is said to p-simulate V_1 if there exists a polynomial time computable function f such that for every $x \in L$, if π_1 is a proof of x for V_1, $f(\pi_1)$ is a proof of x for V_2.

Definition 7. *Effectively p-simulation*

For two proof systems V_1 and V_2 over languages L_1 and L_2, V_2 is said to effectively p-simulate V_1 if there exist polynomial time computable functions f, g such that for every $x_1 \in L_1$, $g(x_1) \in L_2$ and if π_1 is a proof of x_1 for V_1, $f(\pi_1)$ is a proof of $g(x_1)$ for V_2.

In this paper, we are only concerned with effective simulations. The Propositional proof systems we will work with are defined below.

Definition 8. *Cutting Planes*

Let $\Delta = \{A_1 \cdots A_m\}$ be a set of unsatisfiable integer linear inequalities in boolean variables $x_1 \cdots x_n$ of the form $A_j \equiv \sum_i a_{ij} x_i \geq b_i$ where a_{ij} and b_i are integers. A Cutting Planes refutation of Δ is a sequence of lines $B_1 \cdots B_s$ such that B_s is the inequality $1 \geq 0$ and for every $\ell \in \{1 \cdots s\}$ $B_\ell \in \Delta$ or is obtained through one of the following derivation rules for $j,k < \ell$

Addition *From $B_j \equiv \sum_i c_{ij} x_i \geq d_j$ and $B_k \equiv \sum_i c_{ik} x_i \geq d_k$, derive*

$$\sum_i (c_{ij} + c_{ik}) x_i \geq d_j + d_k$$

Multiplication by a constant *From $B_j \equiv \sum_i c_{ij} x_i \geq d_j$, derive*

$$c \sum_i c_{ij} x_i \geq c d_j$$

for an integer $c \geq 0$.

Division by a nonzero constant *From $B_j \equiv \sum_i c_{ij} x_i \geq d_j$ and an integer $c > 0$ such that c divides c_{ij} for all i, derive*

$$\sum_i \frac{c_{ij}}{c} x_i \geq \lceil d_j / c \rceil$$

The semantic version of the system also has the following rule

Semantic inference *If $B_j \equiv \sum_i c_{ij} x_i \geq d_j$, $B_k \equiv \sum_i c_{ik} x_i \geq d_j$ and $B_\ell \equiv \sum_i c_{i\ell} x_i \geq d_j$ are inequalities such that every assignment to $x_1 \ldots x_n$ that satisfies B_j and B_k also satisfies B_ℓ, then from lines B_j and B_k, derive B_ℓ.*

The size of a line is the size of its bit representation. The size of a proof is the sum of sizes of each line. The length of a Cutting Planes proof is equal to the number of lines in the proof. We define the coefficient size of a Cutting Planes proof to be equal to the maximum of the absolute value of all the constants that appear in the proof. $\mathbf{CP}^*$ *is a subsystem of Cutting Planes where the coefficient size is bounded by a polynomial in the number of variables. Without loss of generality, the coefficient size can be bounded by $2^{\mathsf{poly}(l)}$ where l is the length of the proof due to [CCT87].*

The following system is known to simulate SOS and Sherali-Adams.

Definition 9. *Positivestellensatz Calculus/Dynamic SOS [GV01]*

Let $\Gamma = \{P_1 \cdots P_m\}$ *and* $\Delta = \{Q_1 \cdots Q_r\}$ *be two sets of polynomials over* $\mathbb{R}$ *such that the system of equations* $P_1 = 0 \cdots P_m = 0$, $Q_1 \geq 0 \cdots Q_r \geq 0$ *is unsatisfiable. A Dynamic SOS refutation of* Γ, Δ *is a sequence of inequalities* $R_1 \geq 0 \cdots R_s \geq 0$ *where* $R_s = -1$ *and for every* ℓ *in* $\{1 \cdots s\}$, $R_\ell \in \Gamma \cup \Delta$ *or is obtained through one of the following derivation rules for* $j, k < \ell$

1. *From* $R_j = 0$ *and* $R_k = 0$ *derive* $\alpha R_j + \beta R_k = 0$ *for* $\alpha, \beta \in \mathbb{R}$
2. *From* $R_k = 0$ *derive* $x_i R_k = 0$ *for some* $i \in \{1 \cdots n\}$
3. *From* $R_j \geq 0$ *and* $R_k \geq 0$ *derive* $\alpha R_j + \beta R_k \geq 0$ *for* $\alpha \geq 0, \beta \geq 0 \in \mathbb{R}$
4. *From* $R_j \geq 0$ *and* $R_k \geq 0$ *derive* $R_j R_k \geq 0$
5. $R^2 \geq 0$ *for some polynomial* $R \in \mathbb{R}[x_1 \cdots x_n]$

The size of a line is the size of its bit representation. The size of a Dynamic SOS refutation is the sum of sizes of each line of the refutation.

Definition 10. *Polynomial Calculus*

Let $\Gamma = \{P_1 \cdots P_m\}$ *be a set of polynomials in variables* $\{x_1 \cdots x_n\}$ *over a field* $\mathbb{F}$ *such that the system of equations* $P_1 = 0 \cdots P_m = 0$ *has no solution. A Polynomial Calculus refutation of* Γ *is a sequence of polynomials* $R_1 \cdots R_s$ *where* $R_s = 1$ *and for every* ℓ *in* $\{1 \cdots s\}$, $R_\ell \in \Gamma$ *or is obtained through one of the following derivation rules for* $j, k < \ell$

$R_\ell = \alpha R_j + \beta R_k$ *for* $\alpha, \beta \in \mathbb{F}$

$R_\ell = x_i R_k$ *for some* $i \in \{1 \cdots n\}$

The size of the refutation is $\sum_{\ell=1}^{s} |R_\ell|$, *where* $|R_\ell|$ *is the number of monomials in the polynomial* R_ℓ. *The degree of the refutation is* $\max_\ell deg(R_\ell)$.

2.2.3 Generalizations of Polynomial Calculus

We now define a variant of Polynomial Calculus, ΣΠΣ-PC where the proof system is additionally allowed to introduce new variables y_j corresponding to affine forms in the original variables x_i. Thus, each line of the proof is represented by a ΣΠΣ algebraic circuit.

Definition 11. ΣΠΣ-*PC*

Let $\Gamma = \{P_1 \cdots P_m\}$ *be a set of polynomials in variables* $\{x_1 \cdots x_n\}$ *over a field* $\mathbb{F}$ *such that the system of equations* $P_1 = 0 \cdots P_m = 0$ *has no solution. A* ΣΠΣ*-PC refutation of* Γ *is a Polynomial Calculus refutation of a set* $\Gamma' = \{P_1 \cdots P_m, Q_1 \cdots Q_k\}$ *of polynomials over variables* $\{x_1 \cdots x_n\}$ *and* $\{y_1 \cdots y_k\}$ *where* $Q_1 \cdots Q_k$ *are polynomials of the form* $Q_j = y_j - (a_{j0} + \sum_i a_{ij} x_i)$ *for some constants* $a_{ij} \in \mathbb{F}$.

The size of a ΣΠΣ*-PC refutation is equal to the size of the Polynomial Calculus refutation of* Γ'.

We would now like to generalize the above proof system to an arbitrary depth d.

Definition 12. *Depth-d-PC*

Let $d > 2$ *be an integer. Let* $\Gamma = \{P_1 \cdots P_m\}$ *be a set of polynomials in variables* $\{x_1 \cdots x_n\}$ *over a field* $\mathbb{F}$ *such that the system of equations* $P_1 = 0 \cdots P_m = 0$ *has no solution. Let* $S = (y_1 \cdots y_k)$ *be a SLP over* $\{x_1 \cdots x_n\}$ *and* $\mathbb{F}$ *of depth* $d-2$ *defined by* $y_j = Q_j(x_1 \cdots x_n, y_1 \cdots y_{j-1})$. *A Depth-d-PC refutation of* Γ *is a Polynomial Calculus refutation of the set* $\Gamma' = \{P_1 \cdots P_m, y_1 - Q_1, \cdots, y_k - Q_k\}$ *of polynomials over* $\{x_1 \cdots x_n\}$ *and* $\{y_1 \cdots y_k\}$.

The size of a Depth-d-PC refutation is the size of the Polynomial Calculus refutation of Γ'

Although we define the size of a proof in Depth-d-PC in terms of the number of monomials, we will be using the number of lines as a measure of the size, since in our simulations no line contains more than a polynomial number of monomials.

In this work, we will think of d as a large enough constant for all our simulations. A value of $d = 7$ should work for Theorem 2 and $d = 10$ for Theorem 4.

To conclude this section, we state the following result from [RT08a], which is the starting point of our work.

Theorem 0. ***[RT08a]*** *Trinomial-ΠΣ-PC over $\mathbb{Q}$ can simulate syntactic Cutting Planes with the number of lines polynomial in n and the coefficient size.*

2.3 Formal statement of results

We can now restate our results in terms of the proof systems defined in the previous section.

Theorem 1. *ΣΠΣ-PC can p-simulate semantic CP* over $\mathbb{Q}$.*

Theorem 1 is proved in sections 2.4.1 and 2.4.2.

Theorem 2. *Let n and m be positive integers such that $m = O(\mathsf{poly}(\log(n)))$. There is a fixed constant d such that Depth-d-PC over $\mathbb{F}_{p^m}$ can quasipolynomially simulate $\mathsf{AC}^0[q]$-Frege over n variables for any prime q.*

We prove Theorem 2 in sections 2.5.2 and 2.5.2.

Theorem 3. *A TC^0-Frege proof of depth d can be p-simulated by Depth-d'-PC over $\mathbb{F}_m$, where $d' = O(d)$ and m is logarithmic in the size of the largest threshold gate.*

The proof of Theorem 3 is shown in section 2.5.3.

Theorem 4. *There is a fixed constant d such that Depth-d-PC over $\mathbb{F}_{p^{m'}}$ can p-simulate Cutting Planes and Dynamic Sum of Squares, where m' is logarithmic in the maximum number of monomials in any line of the proof.*

Theorem 4 is proved in section 2.5.4.

2.4 Simulations over $\mathbb{Q}$

In this section we outline how we translate inequalities into polynomials over $\mathbb{Q}$, and simulate proofs involving these inequalities into polynomial calculus derivations over their translations.

Consider a line $A_j \equiv \sum_i a_{ij}x_i \geq b_j$ in a CP* proof, where $|a_i|$, $|b|$ are bounded logarithmically in n. We define its translation over $\mathbb{Q}$ as the following

Definition 13. *Translation from CP* to ΣΠΣ-PC*

For a line $A_j \equiv \sum_i a_{ij}x_i \geq b_j$ its translation in ΣΠΣ-PC is defined to be the following pair of lines

$$\prod_{b=0}^{\sum_i a_{ij}^+ - b_j} (y_j - b) = 0$$

$$y_j = \sum_i a_{ij}x_i - b_j$$

In addition, for all i, the equations $x_i(x_i - 1) = 0$ are included in the translation.

That is, we introduce a variable $y_j = \sum_i a_{ij}x_i - b_j$ and indicate the range of values it can take which satisfy the constraint $\sum_i a_{ij}x_i \geq b_j$. For convenience, we will denote by $z \in A$ the equation $\prod_{a\in A}(z-a) = 0$.

The key idea is to note that given two equations $z \in A$ and $z \in B$, we can derive in ΣΠΣ-PC the equation $z \in A \cap B$. We call this the Intersection lemma. A formal proof is provided in Appendix 2.A.1.

2.4.1 Simulating syntactic CP*

We now sketch how all the derivations rules of syntactic CP* can be simulated with the help of the Intersection lemma, concluding Theorem 0. For instance, given equations $y_1 \in A$ and $y_2 \in B$, we derive the range of values a variable $z = y_1 + y_2$ takes as follows. For every $a_1 \in A$,

we derive an equation which states $z \in a_1 + B$ OR $y_1 \in A \setminus \{a_1\}$ where $a_1 + B = \{a_1 + b \mid b \in B\}$. This equation is formally represented as

$$\prod_{c \in a_1 + B} (z - c) \prod_{a \in A \setminus \{a_1\}} (y_1 - a) = 0$$

We can multiply each of these equations by appropriate variables, so that the part about z is the same in all of them. We would now like to eliminate the part about y_1 from these equations. Noting that $\cap_i A \setminus \{a_i\} = \emptyset$, we use the Intersection lemma inductively to eliminate y_1.

For simulating division by an integer c given a variable $z = \sum_i c_i x_i$ and an equation $z \in C$ such that c divides every element of C, we first derive $z \in I$, where I is all possible integer values of the expression $\sum_i c_i x_i$, by using our simulation of addition. We then introduce a variable $z' = z/c$ and from the former equation, we get a set of integer values for z' and from the latter, we get a set of rational values. Using the Intersection lemma now gives the right range for the variable $z' = z/c$.

For a formal proof, see Appendix 2.A.2.

2.4.2 Simulating semantic CP*

In this section we extend the above simulation to include semantic CP*, hence completing the proof of Theorem 1. Let $L_1 \equiv \sum_i a_i x_i \geq d_1$, $L_2 \equiv \sum_i b_i x_i \geq d_2$ be two lines in a Cutting Planes proof and let $L_3 \equiv \sum_i c_i x_i \geq d_3$ be a semantic consequence of L_1 and L_2. Let $y = \sum_i a_i x_i$, $z = \sum_i b_i x_i$ and $w = \sum_i c_i x_i$. Let $A = \{0 \cdots \sum_i a_i^+\}$, $B = \{0 \cdots \sum_i b_i^+\}$ and $C = \{0 \cdots \sum_i c_i^+\}$. Using Lemma 3, we can derive the equations

$$\prod_{a \in A} (y - a) = 0$$

$$\prod_{b \in B} (z - b) = 0$$

$$\prod_{c \in C} (w - c) = 0$$

This restricts the values that can be taken by the tuple (y,z,w) to the three dimensional grid $A \times B \times C$. Let a point (i,j,k) in the grid be *infeasible* if the tuple (y,z,w) never evaluates to it for any assignment to $\{x_i\}$. Our first step is to derive *infeasibility equations* of the form

$$\prod_{\substack{a\in A\\ a\neq i}}(y-a)\prod_{\substack{b\in B\\ b\neq j}}(z-b)\prod_{\substack{c\in C\\ c\neq k}}(w-c)=0$$

which for $(i,j,k) \in A \times B \times C$ tells us that the point (i,j,k) in the grid is infeasible for the tuple (y,z,w).

Lemma 9. *For every infeasible point $(i,j,k) \in A \times B \times C$, ΣΠΣ-PC can derive an infeasibility equation of the above form in $O((\sum_i a_i^+)^2(\sum_i b_i^+)^2(\sum_i c_i^+)^2)$ lines*

The proof of this lemma is left to Appendix 2.A.3.

The next step is to use the ranges of y and z specified in lines L_1 and L_2 to narrow down the possible values that can be taken by w. Our goal will be to get an equation of the form

$$\prod_{c\in C'}(w-c)=0$$

such that each c in C' is feasible for w under the constraints L_1 and L_2 on y and z respectively.

Let P_i be the translation of L_i in Trinomial-ΠΣ-PC, for $i = 1,2,3$. Let $I_{a,b}$ denote the set of all infeasibility equations for points of the form (a,b,k) for some $k \in C$. For an equation P of the form $\prod_{a\in A_1}(y-a)\prod_{b\in B_1}(z-a)\prod_{c\in C_1}(w-a)=0$, denote by $\mathcal{R}_y(P)$ the set A_1, that is the range of values specified by the equation for the variable y. $\mathcal{R}_z$ and $\mathcal{R}_w$ are defined analogously. We describe how to obtain the set C' by the algorithm w-FEASIBLE which operates on the range sets.

Consider a pair $(a,b) \in \mathcal{R}_y(P_1) \times \mathcal{R}_z(P_2)$. For any equation $I \in I_{a,b}$, $\mathcal{R}_w(I)$ gives a list of possible values the variable w can take when $(y,z) = (a,b)$. By Lemma 9, $(y,z,w) = (a,b,c)$ is

```
C' ← ∅
for (a,b) ∈ R_y(P_1) × R_z(P_2) do
    S ← C for I ∈ I_{a,b} do
        S ← S ∩ R_w(I)
    end
    C' ← C' ∪ S
end
return C'
```

Algorithm 1: w-feasible(P_1,P_2)

infeasible if and only if there is an equation $I \in I_{a,b}$ such that $c \notin \mathcal{R}_w(I)$. Therefore, $\bigcap_{I \in I_{a,b}} \mathcal{R}_w(I)$ is precisely the feasible set of values for w, given $(y,z) = (a,b)$. C' is the union of such sets over all possible pairs $(a,b) \in \mathcal{R}_y(P_1) \times \mathcal{R}_z(P_2)$ and hence is the set of all feasible values of w.

This algorithm over range sets can be easily translated to a proof of $\prod_{c \in C'}(w-c) = 0$ from P_1 and P_2 in Trinomial-ΠΣ-PC as follows. To simulate the inner **for** loop, we use the Intersection lemma inductively over all equations in $I_{a,b}$ to get equations $J_{a,b}$ such that $\mathcal{R}_w(J_{a,b}) = \bigcap_{I \in I_{a,b}} \mathcal{R}_w(I)$. Note that $\mathcal{R}_y(J_{a,b}) = A \setminus \{a\}$ and $\mathcal{R}_z(J_{a,b}) = B \setminus \{b\}$. Thus using the Intersection lemma again inductively over the set $\{J_{a,b}\}$ analogous to Lemma 6 would give an equation free of y and z, where w ranges over $\bigcup_{(a,b)} \mathcal{R}_w(J_{a,b})$. Any semantic consequence P_3 must be such that $\mathcal{R}_w(P_3) \supseteq C'$ and hence is easily derived.

2.5 Simulations over $\mathbb{F}_{p^m}$

2.5.1 Simulating syntactic CP*

We now carry out the simulation in Section 2.4.1 in Depth-d-PC over a large enough field extension F_{p^m} of a finite field F_p. This will be of use in the next section, where we simulate $AC^0[p]$-Frege in Depth-d-PC over F_{p^m}. For the following discussion, we set $d = 5$

To represent large integers over $\mathbb{F}_{p^m}$, we choose a primitive element α and for a boolean x_i perform the linear transformation $y_i = 1 + (\alpha - 1)x_i$. Since x_i is boolean, y_i is essentially

equivalent to the mapping $x_i \mapsto \alpha^{x_i}$. The expression $\sum_i a_i x_i$ is thus represented as $\alpha^{\sum_i a_i x_i}$. The goal here is to show that all the steps of the simulation in section 2.4.1 can still be performed after this transformation.

Theorem 7. *Depth-d-PC over F_{p^m} can simulate syntactic Cutting Planes with the number of lines polynomial in n and the coefficient size, where m is logarithmic in n and the coefficient size.*

Let s_1 be the coefficient size of the Cutting Planes proof. Define $s = ns_1$. Choose m to be the smallest integer such that $2s^2 < p^m - 1$. Let α be an arbitrary primitive element of $\mathbb{F}_{p^m}$.

Definition 14. *Translation of Cutting Planes to Depth-d-PC over $\mathbb{F}_{p^m}$*

Given a line $\sum_i a_i x_i \geq b_i$ in Cutting Planes, the translation of the above line is defined as the following lines, where y_i and y are new variables.

$$y_i = (\alpha^{a_i} - 1)x_i + 1$$

$$y = \prod_i y_i$$

$$(y - \alpha^{b_i})(y - \alpha^{b_i+1}) \cdots (y - \alpha^{\sum_i a_i^+}) = 0$$

An integer c such that $0 \leq c \leq s$ is represented as α^c, whereas for $-s \leq c < 0$ we represent it as $\alpha^{-|c|} \equiv \alpha^{(p^m-1)-|c|}$. Since $2s \leq 2s^2 < p^m - 1$, these representations are unique.

The technical details of the simulating the rules of CP are largely similar to that over $\mathbb{Q}$ and are hence left to Appendix 2.A.4

2.5.2 Simulating $\mathsf{AC}^0[q]$-Frege

Case of $q = p$

For the purpose of this section, we set $d = 7$. We will use the simulation of $\mathsf{AC}^0[p]$-Frege in [MP98] to show that the same can be carried out in Depth-d-PC over $\mathbb{F}_{p^m}$. We fix m to be

a large enough integer such that $m = O(\mathsf{poly}(\log(n)))$, so that the field we are working over is quasipolynomially sized. Below we describe the proof system of [MP98] and their simulation of $\mathsf{AC}^0[p]$-Frege.

The Proof System of Maciel and Pitassi Maciel and Pitassi [MP98] define a proof system with mod p, negation, AND, OR and threshold connectives, based on the system PTK by Buss and Clote [BC96] which we describe below.

Connectives Let $x_1 \cdots x_n$ be boolean variables. For $0 \leq j < p$, let $\oplus_j^p(x_1 \cdots x_n)$ denote the connective which is 1 if and only if $\sum_i x_i = j \mod p$. For any integer t, let $Th_t(x_1 \cdots x_n)$ denote the connective which is 1 if and only if $\sum_i x_i \geq t$. Let $\wedge(x_1 \cdots x_n)$, $\vee(x_1 \cdots x_n)$ denote AND and OR connectives of arity n and $\neg$ denote the NOT gate.

Formulas A *formula* is recursively defined as follows. Input variables $x_1 \cdots x_n$ are formulas of size 1 and depth 1. A formula φ is an expression of the form $g(\varphi_1 \cdots \varphi_k)$, where g is any of the connectives described above and $\varphi_1 \cdots \varphi_k$ are formulas. The $depth(\varphi)$ is defined as $\sum_{i=1}^k depth(\varphi_i) + 1$. The $size(\varphi)$ is defined as $\sum_{i=1}^k size(\varphi_i) + k + 1$ if g is not a threshold connective, and it is defined as $\sum_{i=1}^k size(\varphi_i) + t + k + 1$ if g is a threshold connective of the form $Th_t(\varphi_1 \cdots \varphi_k)$.

Cedents and Sequents A cedent Γ is defined as a sequence of formulas $\varphi_1 \cdots \varphi_k$. We will use capital Greek letters to denote cedents. A sequent is an expression of the form $\Gamma \to \Delta$, where Γ and Δ are cedents. The interpretation of a sequent is that the AND of all the formulas in Γ implies the OR of all the formulas in Δ. The size and depth of a cedent are respectively the sum of sizes and the maximum of depths of all the formulas in it. The size of a sequent is the sum of sizes of both cedents, and the depth is the maximum of the depths of both cedents.

Definition of a Proof A proof in this system is defined as a sequence of sequents $\mathcal{S}_1 \cdots \mathcal{S}_m$ such that each $\mathcal{S}_i$ is either an initial sequent, or is derived from sequents $\mathcal{S}_j$ for $j < i$ through one of the rules listed below. The size and depth of a proof are respectively the sum of sizes and the maximum of depths of all sequents in it.

The initial sequents and the derivation rules are listed below.

The proof system of Maciel and Pitassi [MP98]

initial sequents

1. $\varphi \to \varphi$ for any formula φ
2. $\to \wedge()\ ;\ \vee() \to$
3. $\oplus^p_j() \to$ for $1 \leq j < p\ ;\ \to \oplus^p_0()$
4. $Th_t() \to$
5. $\to Th_0(\varphi_1 \cdots \varphi_k)$ for any $k \geq 0$

structural rules

weakening: $\dfrac{\Gamma, \Delta \to \Gamma'}{\Gamma, \varphi, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \Delta'}{\Gamma \to \Gamma', \varphi, \Delta'}$

contract: $\dfrac{\Gamma, \varphi, \varphi, \Delta \to \Gamma'}{\Gamma, \varphi, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \varphi, \varphi, \Delta'}{\Gamma \to \Gamma', \varphi, \Delta'}$

permute: $\dfrac{\Gamma, \varphi_1, \varphi_2, \Delta \to \Gamma'}{\Gamma, \varphi_2, \varphi_1, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \varphi_1, \varphi_2, \Delta'}{\Gamma \to \Gamma', \varphi_2, \varphi_1, \Delta'}$

cut rule

$$\frac{\Gamma, \varphi \to \Delta \qquad \Gamma' \to \varphi, \Delta'}{\Gamma, \Gamma' \to \Delta, \Delta'}$$

logical rules

$\neg$: $\dfrac{\Gamma \to \varphi, \Delta}{\neg\varphi, \Gamma \to \Delta} \quad \dfrac{\varphi, \Gamma \to \Delta}{\Gamma \to \neg\varphi, \Delta}$

$\wedge$-left: $\dfrac{\varphi_1, \wedge(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{\wedge(\varphi_1 \cdots \varphi_k), \Gamma \to \Delta}$

$\wedge$-right: $\dfrac{\Gamma \to \varphi_1, \Delta \qquad \Gamma \to \wedge(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \wedge(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

$\vee$-left: $\dfrac{\varphi_1, \Gamma \to \Delta \qquad \vee(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{\vee(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

$\vee$-right: $\dfrac{\Gamma \to \varphi_1, \vee(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \vee(\varphi_1 \cdots \varphi_k), \Delta}$

$\oplus_i$-left: $\dfrac{\varphi_1, \oplus^p_{i-1}(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta \qquad \oplus^p_i(\varphi_2 \cdots \varphi_k), \Gamma \to \varphi_1, \Delta}{\oplus^p_i(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

$\oplus_i$-right: $\dfrac{\varphi_1, \Gamma \to \oplus^p_{i-1}(\varphi_2 \cdots \varphi_k), \Delta \qquad \Gamma \to \varphi_1, \oplus^p_i(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \oplus^p_i(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

Th_t-left: $\dfrac{Th_t(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta \qquad \varphi_1, Th_{t-1}(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{Th_t(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

Th_t-right: $\dfrac{\Gamma \to \varphi_1, Th_t(\varphi_2 \cdots \varphi_k), \Delta \qquad \Gamma \to Th_{t-1}(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to Th_t(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

Translating lines We will now define translations of lines in the above proof system. For a formula φ, we denote its translation in Depth-d-PC by $tr(\varphi)$. Let $x_1 \cdots x_n$ be the variables of the original proof. Below we list the translations for a formula built with each connective. The interpretation is that for any formula φ, $tr(\varphi) = 0$ if and only if φ is true.

$$tr(x_i) = 1 - x_i$$

$$tr(\vee(\varphi_1 \cdots \varphi_k)) = \prod_i (tr(\varphi_i))$$

$$tr(\wedge(\varphi_1 \cdots \varphi_k)) = 1 - \prod_i tr(\neg\varphi_i)$$

$$tr(\oplus^p_i(\varphi_1 \cdots \varphi_k)) = (\textstyle\sum_{j=1}^{k} \varphi_j - i)^{p-1} \text{ for } 0 \leq i < p$$

$tr(Th_t(\varphi_1 \cdots \varphi_k)) = (y - \alpha^t) \cdots (y - \alpha^k)$

where $y = \prod_i ((\alpha - 1) tr(\neg \varphi_i) + 1)$

$tr(\neg \varphi) = 1 - tr(\varphi)$ if φ does not contain a Th_t connective

$tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = (y - 1) \cdots (y - \alpha^{t-1})$

where $y = \prod_i ((\alpha - 1) tr(\neg \varphi_i) + 1)$, for $t \geq 1$

The translation $tr(\mathcal{S})$ of a sequent $\mathcal{S}$ of the form $\varphi_1 \cdots \varphi_k \rightarrow \varphi'_1 \cdots \varphi'_m$ is given by the equation

$$\prod_{i=1}^{k} tr(\neg \varphi_i) \prod_{j=1}^{m} tr(\varphi'_j) = 0$$

Note that the translations of all the connectives except the threshold connective take only boolean values over $\mathbb{F}_{p^m}$.

Simulating proofs We now describe the connection between $\mathsf{AC}^0[p]$-Frege and the proof system of Maciel and Pitassi. By the following theorem of Allender [All89], any $\mathsf{AC}^0[p]$ circuit can converted to a depth three circuit of a special form.

Theorem 8. ***[All89]***

Any $\mathsf{AC}^0[p]$ *circuit can be converted to a quasipolynomial sized depth three circuit with an unweighted Threshold gate at the top,* MOD_p *gates of quasipolynomial fan-in in the middle and* $\wedge$ *gates of polylogarithmic fan-in at the bottom*

Depth three circuits with an unweighted Threshold, $\wedge$ or $\vee$ gate at the top, MOD_p gates in the middle and $\wedge$ gates of polylogarithmic fan-in in the size of the circuit at the bottom are referred to as *flat circuits* by [MP98]. For an $\mathsf{AC}^0[p]$ circuit φ, its *flattening* $fl(\varphi)$ is defined as the flat circuit given by the above theorem. Proofs in $\mathsf{AC}^0[p]$-Frege can be thought of as a list of sequents such that every formula that appears in each of them is an $\mathsf{AC}^0[p]$ circuit. For a sequent $\varphi_1 \cdots \varphi_k \rightarrow \varphi'_1 \cdots \varphi'_m$ that appears in a $\mathsf{AC}^0[p]$-Frege proof, we can define a flattening of

the sequent $fl(\varphi_1)\cdots fl(\varphi_k) \to fl(\varphi'_1)\cdots fl(\varphi'_m)$ in the proof system of Maciel and Pitassi. A *flat proof* of such a sequent is such that every formula that appears in the proof is a flat circuit. The simulation theorem of [MP98] states the following

Theorem 9. ***[MP98]***

Let $\mathcal{S}$ be a sequent which has a depth d proof in $\mathsf{AC}^0[p]$-Frege. Then its flattening $fl(\mathcal{S})$ has a flat proof of size $2^{(\log n)^{O(d)}}$ in the proof system of Maciel and Pitassi.

We will show that flat proofs can be simulated in Depth-d-PC by showing the following

Theorem 10. *Let $\mathcal{S}$ be a sequent which has a flat proof of size s in the proof system of Maciel and Pitassi. Then there is a proof of the equation $tr(\mathcal{S})$ in Depth-d-PC from the equations $x_i(x_i-1)=0$ with $poly(s)$ lines.*

To prove the above theorem, it is sufficient to show that for each rule that derives a sequent $\mathcal{S}_3$ from sequents $\mathcal{S}_1$ and $\mathcal{S}_2$, there is a derivation of the equation $tr(\mathcal{S}_3)$ from the equations $tr(\mathcal{S}_1)$, $tr(\mathcal{S}_2)$ and $x_i(x_i-1)=0$ in Depth-d-PC. The details of how each such rule can be simulated are left to Appendix 2.B.1

Case of $q \neq p$

We now extend the simulation of the previous section to show that $\mathsf{AC}^0[q]$-Frege can be simulated in Depth-d-PC over F_{p^m}, for distinct primes p and q, hence proving Theorem 2. Using the theorem of Maciel and Pitassi (Theorem 14 above) for $\mathsf{AC}^0[q]$-Frege, we obtain a flat proof with $\oplus_i^q$ connectives. To simulate it, we can reuse the lemmas of the previous section, except for the $\oplus_i^q$ connectives. To define their translation, choose m such that $q \mid p^m-1$ and let $r=(p^m-1)/q$. The translation is now defined as

$$tr(\oplus_i^q(\varphi_1\cdots\varphi_k)) = \left((y-\alpha^{ir})\right)^{p^m-1}$$

where $y = \prod_i((\alpha^r - 1)tr(\neg\varphi_i) + 1)$ and $tr(\neg \oplus_i^q (\varphi_1 \cdots \varphi_k)) = 1 - tr(\oplus_i^q(\varphi_1 \cdots \varphi_k))$

Simulating the rules is similar to the previous section. The proof for one such rule is shown in Appendix 2.B.2

2.5.3 Simulating TC^0-Frege

In this section, we show that a TC^0-Frege proof of depth d_0 can be transformed into a Depth-d-PC proof over $\mathbb{F}_{p^m}$, where $d = O(d_0)$, proving Theorem 3. In the previous section we translated $Th_t(\varphi_1 \cdots \varphi_k)$ as

$$tr(Th_t(\varphi_1 \cdots \varphi_k)) = (y - \alpha^t) \cdots (y - \alpha^k)$$

$$tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = (y - 1) \cdots (y - \alpha^{t-1})$$

where $y = \prod_i((\alpha - 1)tr(\neg\varphi_i) + 1)$. Clearly this translation requires $tr(\varphi_i)$ to be boolean and can itself take non-boolean values. Since there is only one top Threshold gate in a flat circuit, the formulae φ_i were threshold free and thus $tr(\varphi_i)$ only took on boolean values. But in a TC^0-Frege proof, the formulae φ_i can themselves contain Threshold gates and thus $tr(\varphi_i)$ may be non-boolean. To fix this problem, we redefine the translation of a Threshold gate to be the following, essentially forcing it to be boolean.

$$tr(Th_t(\varphi_1 \cdots \varphi_k)) = \left((y - \alpha^t) \cdots (y - \alpha^k)\right)^{p^m - 1}$$

where $y = \prod_i((\alpha - 1)tr(\neg\varphi_i) + 1)$ and $tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = 1 - tr(Th_t(\varphi_1 \cdots \varphi_k))$.

It is easy to generalize Lemma 13 to derive the fact that the above translation only takes boolean values. Now, note that any rule other than the Th_t is unaffected by this new translation since it only assumes that its arguments are boolean and hence we can use the lemmas of the previous section directly. However, simulation of the Th_t rule relies on the old translation. To

bridge the gap, we only need to show that the old and new translations of Th_t and $\neg Th_t$ are interchangeable within the proof system. The following lemmas are proved in Appendix 2.C

Lemma 1. *Given the equation*

$$\left((y-\alpha^t)\cdots(y-\alpha^k)\right)^{p^m-1}=0$$

we can derive

$$(y-\alpha^t)\cdots(y-\alpha^k)=0$$

and vice versa.

Lemma 2. *Given the equation*

$$1-\left((y-\alpha^t)\cdots(y-\alpha^k)\right)^{p^m-1}=0$$

we can derive

$$(y-1)\cdots(y-\alpha^{t-1})=0$$

and vice versa.

Existence of Feasible Interpolation

Bonet, Pitassi and Raz [BPR00] have shown that TC^0-Frege does not have feasible interpolation unless Blum integers can be factored by polynomial sized circuits. By the above simulation, we can state the following

Theorem 11. *Depth-d-PC does not have feasible interpolation unless Blum integers can be factored by polynomial sized circuits*

2.5.4 Dealing with large coefficients

It is well-known that arbitrary threshold gates can be simulated by simple majority gates of higher depth. In particular, a tight simulation was proven by Goldmann, Hastad and Razborov [GHR92] who show that depth $d+1$ TC^0 circuits are equivalent to depth d threshold circuits with arbitrary weights. However, the analogous result has not been proven in the propositional proof setting. In order to simulate arbitrary weighted thresholds in our low depth extension of PC, we will we use a different simulation of high weight thresholds by low weight ones.

The basic idea will be to use simple, shallow formulas that compute the iterated addition of n binary numbers, each with $m = poly(n)$ bits [MT98]. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$ be the set of n binary numbers, each of length $m = poly(n)$, where $\mathbf{a_i} = a_{i,m}, \ldots, a_{i,1}$. We will break up the m coordinates into $m/\log m$ blocks, each of size $\log m$; let $L_j(\mathbf{a_i})$ denote the j^{th} block of $\mathbf{a_i}$. The high level idea is to compute the sum by first computing the sum *within* each block, and then to combine using carry-save-addition.

In more detail, let $\mathbf{a_i^o}$ denote the "odd" blocks of $\mathbf{a_i}$ – so $\mathbf{a_i^o}$ consists of $m/\log m$ blocks, where for j odd, the j^{th} block is $L_j(\mathbf{a_i})$, and for j even, the j^{th} block is all zeroes (and similarly, $\mathbf{a_i^e}$ denotes the even blocks of $\mathbf{a_i}$). Let S^o be equal to $\sum_{i\in[n]} \mathbf{a_i^o}$, and similarly let S^e be equal to $\sum_{i\in[n]} \mathbf{a_i^e}$. We will give a SLP for computing the bits of S^o and S^e and then our desired sum, $S^o + S^e$, is obtained using the usual carry-save addition which can be computed by a depth-2 SLP. The main point is that we have padded $\mathbf{a_i^o}$ and $\mathbf{a_i^e}$ with zeroes in every other block; this enables us to compute S^o (and similarly S^e) *blockwise* (on the odd blocks for S^o and on the even blocks for S^e), because no carries will spill over to the next nonzero block. Then since the blocks are very small ($\log m$ bits), the sum within each block can be carried out by brute-force.

Our construction below generalizes this to the case where the $\mathbf{a_i}$'s are not large coefficients, but instead they are the product of a monomial and a large coefficient. After formally describing this low-depth representation, it remains to show how to efficiently reason about these low-depth representations in order to carry out the rule-by-rule simulation of general Cutting Planes and

SOS.

2.5.5 Bit vector representations of CP/SOS proof lines

Definition 15. *Derivations in Depth-d-PC*

Let m_0 be an upper limit on the number of monomials in any polynomial we wish to represent. We work over an arbitrary field larger than m_0^2.

To indicate that a new extension variable y_i is being introduced and set to a value a_i, we write

$$y_i := a_i$$

To indicate that a line $P = 0$ in Depth-d-PC can be derived from $P_1 = 0$, $P_2 = 0$, $\cdots$,$P_k = 0$, we write

$$P_1, P_2, \cdots, P_k \vdash P$$

To indicate that a line $P = 0$ can be derived just from the axioms of the form $x_i^2 = x_i$ for all boolean variables x_i, we write

$$\vdash P$$

Below we formally define the representation of binary numbers as bit vectors.

Definition 16. *Bit vectors*

We represent an integer using its bit representation by introducing a variable for each of its bits. Let a be an integer with bits $a_m \cdots a_1$. A bit vector $\mathbf{a} = [a_m \cdots a_1]$ representing the integer a in our system is a set of auxiliary variables $y_m \cdots y_1$ such that $y_i := a_i$. Define $\mathbf{a}(i) = y_i = a_i$. Integers which are represented as vectors are written in boldface.

Let m_0 be an upper limit on the number of monomials in any polynomial we wish

to represent and let m_1 be an upper limit on any coefficient we wish to represent. Set $m = 10\lceil log(m_0) + log(m_1)\rceil$. The bit vectors in this simulation will all be of dimension m, i.e. all integers we represent will be of at most m bits. Any vector of dimension $> m$ generated in any operation is automatically truncated to dimension m by dropping the higher order bits.

The bit representation chosen is Two's complement. That is, a positive integer is represented in binary in the usual way. Let b be a positive integer represented by $\mathbf{b}$*. Let* $\mathbf{b}_1$ *be the vector obtained by flipping all the bits in* $\mathbf{b}$*. Then we define the vector* $-\mathbf{b}$ *as* $\mathbf{b}_1 \oplus \mathbf{1}$*, where* $\oplus$ *operation on vectors, defined below, simulates the usual bitwise addition operation and* $\mathbf{1}$ *is the vector representation of the integer 1.* $\mathbf{0}$*, the all zeros vector, represents the integer 0. For any vector* $\mathbf{a}$*,* $\mathbf{a}(m)$ *is the sign bit of* $\mathbf{a}$*.* $\mathbf{a}$ *is said to be negative if the sign bit is one.*

In order to make correct computation using the above Two's complement representation of binary numbers, we need to ensure that the bit length of all numbers represented is bounded. We therefore define the *length* of a vector in our simulation, and later show that such vectors are of bounded length.

Definition 17. *Length of a vector*
The length of a non-negative vector $\mathbf{a}$ *is the highest index i such that* $\mathbf{a}(i) \neq 0$ *and zero if such an i does not exist. The length of a negative vector* $\mathbf{b}$ *is the highest index i such that* $\mathbf{b}(i) \neq 1$*. Equivalently, the length of a vector* $\mathbf{a}$ *is the highest index i such that* $\mathbf{a}(i) \neq \mathbf{a}(m)$*.*

We now define the usual addition operation for binary numbers, over their vector representations. Since we work in a low depth setting, we need to use *Carry-Save* addition to represent the sum and carry bits.

2.5.6 Operations on bit vectors

Definition 18. *The Bitwise Addition operation* $\oplus$
We define below the operator on vectors corresponding to the usual carry-save addition. For

two bits y and z, let $y \oplus z$ represent the XOR of the bits. Given two bit vectors $\mathbf{y} = [y_m \cdots y_1]$ and $\mathbf{z} = [z_m \cdots z_1]$, the bitwise addition operation $\mathbf{y} \oplus \mathbf{z}$ produces a vector $[w_{m+1} \cdots w_1]$ such that

$$w_i := y_i \oplus z_i \oplus c_i$$

for $i \leq m$ and $w_{m+1} = c_m$ where

$$c_i := \vee_{j<i}(y_j \wedge z_j \wedge_{j<k<i} (y_k \oplus z_k))$$

for $1 < i \leq m$ and $c_1 = 0$.

c_i are referred to as the carry bits in $\mathbf{y} \oplus \mathbf{z}$

Monomial terms a_1X_1 in our system are represented by a "scalar multiplication" of X_1 with the vector $\mathbf{a}_1$, which we define below.

Definition 19. *Scalar multiplication*
For a bit z and a vector $\mathbf{y}$, let $z\mathbf{y} = \mathbf{y}z$ represent the vector obtained by multiplying every bit of $\mathbf{y}$ by z.

In order to represent a line $a_1X_1 + \cdots + a_nX_n - a_0 \geq 0$ in Cutting Planes, we define an operation $\mathcal{S}$ over the vectors $\mathbf{a}_1X_1, \cdots, \mathbf{a}_nX_n$ such that the resultant vector is a representation of $a_1X_1 + \cdots + a_nX_n - a_0$ and has a low depth in $X_1, \cdots, X_n$. This uses the idea of representing high weight thresholds using low depth majority gates described earlier.

Definition 20. *The Set Addition operation $\mathcal{S}(.)$*
We will now define the representation of the bitwise addition of vectors $\mathbf{a}_1X_1, \cdots, \mathbf{a}_tX_t$, where $\mathbf{a}_1, \cdots, \mathbf{a}_t$ are integer constants and $X_1, \cdots, X_t$ are monomials.

Let $m_2 = \lceil m/\log(m_0) \rceil$. For a constant $\mathbf{a}$, partition the bits of $\mathbf{a}$ into m_2 blocks of length at most $\log(m_0)$. Let $L_j(\mathbf{a})$, $j \in [m_2]$ denote the j^{th} block of bits, so that the bits of $\mathbf{a}$ can be obtained by a concatenation of the bits $L_{m_2}(\mathbf{a})...L_1(\mathbf{a})$. Since $L_j(\mathbf{a})$ is only $\log(m_0)$ bits long, its magnitude

is at most m_0. Let $[L_j(\mathbf{a})]$ refer to the integer represented by the vector $L_j(\mathbf{a})$. Define $\mathbf{a}^o$ to be the vector obtained by replacing all even numbered blocks of $\mathbf{a}$ with zeroes. $\mathbf{a}^e$ is analogously defined by zeroing out the odd numbered blocks. For monomials $X_1 \cdots X_t$ and $t < m_0$, we would like to define bit vectors $\mathcal{S}^o(\mathbf{a}_1 X_1, \cdots, \mathbf{a}_t X_t)$ and $\mathcal{S}^e(\mathbf{a}_1 X_1, \cdots, \mathbf{a}_n X_t)$ to be the bit representations of the polynomials $\sum_{i=1}^t a_i^o X_i$ and $\sum_{i=1}^t a_i^e X_i$. We accomplish this using constant depth SLPs as follows.

We define a constant depth SLP to compute the k^{th} bit of the j^{th} block of $\mathcal{S}^o$, represented by $L_{jk}(\mathcal{S}^o)$. The important observation is that we can compute $\mathcal{S}^o$ two blocks at a time since for odd j, $\sum_i [L_j(\mathbf{a}_i^o)] X_i$ is at most m_0^2 and thus can be represented by $2\log(m_0)$ bits or exactly two blocks. Let C_ℓ be the set of integers in $[m_0^2]$ such that the ℓ^{th} bit of their binary representation is one. Then for odd j, $L_{jk}(\mathcal{S}^o)$ is one if and only if

$$\prod_{\beta \in C_k} \left(\sum_i [L_j(\mathbf{a}_i^o)] X_i - \beta \right) = 0$$

and for even j, $L_{jk}(\mathcal{S}^o)$ is one if and only if

$$\prod_{\beta \in C_{\log(m_0)+k}} \left(\sum_i [L_{j-1}(\mathbf{a}_i^o)] X_i - \beta \right) = 0$$

Therefore, the bit $L_{jk}(\mathcal{S}^o)$ can be represented as a constant depth SLP of size $O(m_0)$ by representing the left hand side of the above equations as a SLP over a field of characteristic larger than m_0^2, similar to the simulation of CP in the earlier sections, and then raising the result of that SLP to the order of the multiplicative group that we are working in. The bits of $\mathcal{S}^e$ are represented analogously.*

The operation $\mathcal{S}$ over vectors $\mathbf{a}_1 X_1, \cdots, \mathbf{a}_t X_t$ now defined as

$$\mathcal{S}^o(\mathbf{a}_1 X_1, \cdots, \mathbf{a}_t X_t) \oplus \mathcal{S}^e(\mathbf{a}_1 X_1, \cdots, \mathbf{a}_n X_t)$$

.

2.5.7 Representing a line from CP/SOS in Depth-d-PC

We now define the translation of a line $a_1X_1 + \cdots + a_nX_k - a_0 \geq 0$ in Cutting Planes/SOS, where $X_1 \ldots X_k$ are monomials.

Definition 21. *Representing an inequality*

Let $P = a_1X_1 + \cdots + a_kX_k$ be a polynomial where the X_i are monomials. Then the line $P \geq 0$ is represented as

$$\mathcal{S}(\mathbf{a}_1X_1, \cdots, \mathbf{a}_kX_k)(m) = 0$$

and $P = 0$ is represented as

$$\mathcal{S}(\mathbf{a}_1X_1, \cdots, \mathbf{a}_kX_k) = \mathbf{0}$$

Let $\mathcal{R}(P)$ denote the vector $\mathcal{S}(\mathbf{a}_1X_1, \cdots, \mathbf{a}_kX_k)$.

2.5.8 Simulating Cutting Planes

Addition

Before we prove the simulation for addition, we need the following key properties of the vector representation. They are proved in Section 2.D.

The lemma below states that our system can prove the associativity of the operation $\oplus$ over vectors.

Lemma 24. *For any three bit vectors* $\mathbf{y}$, $\mathbf{z}$ *and* $\mathbf{w}$

$$\vdash (\mathbf{y} \oplus \mathbf{z}) \oplus \mathbf{w} - \mathbf{y} \oplus (\mathbf{z} \oplus \mathbf{w})$$

We then need to be able to interchangeably use the operations $\mathcal{S}$ and $\oplus$ for vector addition

Lemma 26. $\vdash \mathcal{S}(\mathbf{y}_1, \cdots, \mathbf{y}_i) - \mathcal{S}(\mathbf{y}_1, \cdots, \mathbf{y}_{i-1}) \oplus \mathbf{y}_i$

We then extend this to show that the vector representation of the sum of two lines is the $\oplus$ of the vector representations of each line.

Lemma 28. *Let P and Q be two polynomials. Then $\mathcal{R}(P+Q) = \mathcal{R}(P) \oplus \mathcal{R}(Q)$.*

This concludes simulation of the addition rule.

Multiplication by a constant

In order to simulate multiplication by a power of two, we left-shift bits of the corresponding bit vector by the required amount, and add zero bits at the end. Multiplication by any constant can then be simulated by the above in combination with the Addition rule.

Division by a constant

To simulate the division rule in Cutting Planes we use the following lemma.

Lemma ??. *Let $P = a_1x_1 + \cdots + a_nx_n - a_0$ where a_i are non-negative, $a_1 \cdots a_n$ are even and a_0 is odd. Then we can derive*

$$\mathcal{R}(P)(m) \vdash \mathcal{R}(P-1)(m)$$

We can now simulate the division rule by using the above lemma and then dropping the last bit of the vector $\mathcal{R}(P-1)$ (which would be zero).

2.5.9 Simulating Dynamic SOS

Rules 1, 2 and 3 of Definition 9 follow from the above simulation of Cutting Planes.

Multiplication of two lines

To simulate the multiplication rule of SOS, we need to define an operation which, given the vectors $\mathbf{a}_1$ and $\mathbf{b}_1$, produces a vector that is equivalent to the representation of a_1b_1. We define it as a shifted sum based on the grade school algorithm for binary multiplication.

Definition 22. *Shifted sum*

For a vector $\mathbf{y}$*, let* $2^k\mathbf{y}$ *denote the vector obtained by shifting the bits of* $\mathbf{y}$ *to the left by k positions, and padding the least significant k positions with zeros. Given two vectors* $\mathbf{y}$ *and* $\mathbf{z} = [z_{m-1} \cdots z_0]$ *, the shifted sum of* $\mathbf{y}$ *and* $\mathbf{z}$ *is defined as the vector*

$$\mathcal{SS}(\mathbf{y}, \mathbf{z}) = \mathcal{S}(z_0\mathbf{y}, \cdots, z_{m-1}2^{m-1}\mathbf{y})$$

We then show that our system can prove that the vector obtained by using this operation is indeed what we want.

Lemma 40. *Let P and Q be two polynomials, represented by bit vectors* $\mathbf{y}_0$ *and* $\mathbf{z} = [z_{m-1} \cdots z_0]$*, with at most* m_0 *monomials and coefficients bounded by* m_1 *in absolute value. Then,*

$$\vdash \mathcal{R}(PQ) - \mathcal{SS}(\mathbf{y}_0, \mathbf{z})$$

We now need to show that the as long as P and Q have coefficients not exceeding bit length m, we can derive from $\mathcal{R}(P)(m) = 0$ and $\mathcal{R}(Q)(m) = 0$ the lines $\mathcal{R}(P+Q)(m) = 0$ and $\mathcal{R}(PQ)(m) = 0$. It is an easy observation that if the bit lengths of the coefficients in P and Q are bounded, then the vectors $\mathcal{R}(P)$ and $\mathcal{R}(Q)$ are of bounded length. Thus it suffices to show the following.

Lemma 34. *For any two vectors* $\mathbf{a}$ *and* $\mathbf{b}$ *of length at most* $\ell < m - 1$

$$\mathbf{a}(m), \mathbf{b}(m) \vdash (\mathbf{a} \oplus \mathbf{b})(m)$$

Lemma 35. *Let* $\mathbf{y}$ *and* $\mathbf{z}$ *be two non-negative vectors of length* ℓ *such that* $3\ell < m - 1$*. Then*

$$\mathbf{y}(m), \mathbf{z}(m) \vdash \mathcal{SS}(\mathbf{y}, \mathbf{z})(m)$$

This completes the simulation of the rule which takes the product of two lines in SOS.

Squaring rule

To simulate the rule in SOS which introduces a line $P^2 \geq 0$ for any polynomial P, we need the following lemmas.

The lemma below states that if the sign bit of $\mathbf{y}$ is one, then the sign bit of $-\mathbf{y}$ is zero.

Lemma 30. *For any vector* $\mathbf{y}$ *of length* $\ell < m-1$,

$$\mathbf{y}(m) - 1 \vdash (-\mathbf{y})(m)$$

The following lemma shows that for a vector representing a polynomial P, the negation of it represents the polynomial $-P$.

Lemma 31. *Let P be a polynomial represented by a vector* $\mathbf{y}$*. Then* $\vdash \mathcal{R}(-P) - (-\mathbf{y})$.

The rule which derives $P^2 \geq 0$ can now be easily simulated by branching on the sign bit of the vector $\mathcal{R}(P)$. Assuming it to be zero, we can use Lemma 35 to derive $\mathcal{R}(P^2)(m) = 0$. In the other case, we can use Lemma 30 and Lemma 31 to derive that the sign bit of $\mathcal{R}(-P)$ is zero. We can now use Lemma 35 again to derive $\mathcal{R}(P^2)(m) = 0$.

2.5.10 Concluding the simulation

By simulating any refutation in Cutting Planes/SOS rule by rule using the above lemmas, we end up with the representation of the line $-1 \geq 0$ i.e.

$$\mathcal{R}(-1)(m) = 0$$

Since -1 is represented by the all ones vector, this gives a contradiction.

Appendix

2.A Small-weight Cutting Planes Simulations

Notational Remark In Depth-d-PC, we sometimes use "inline" definitions to indicate the new variables y_j introduced. For instance, the equation

$$x_1(x_1+1)=0$$

represents the equations

$$x_1 y_1 = 0$$

$$y_1 = x_1 + 1$$

Thus when we refer to the monomial corresponding to $x_1(x_1+1)$, we are referring to $x_1 y_1$.

Though ΣΠΣ-PC captures the effect of size reductions due to allowing linear transformations within the proof, it turns out that it is more powerful than required for our simulation in Theorem 1, so we define the tightest restriction of it where we can still do the simulation.

Definition 23. *A Trinomial is a polynomial with at most three monomials*

Definition 24. *Trinomial-*ΠΣ*-PC*

Let $\Gamma = \{P_1 \ \cdots \ P_m\}$ *be a set of polynomials over a field* $\mathbb{F}$ *such that each* $P \in \Gamma$ *is either an affine*

form or a trinomial in $\{x_1 \cdots x_n\}$. *Let the system of equations* $P_1 = 0 \cdots P_m = 0$ *have no solution. Let* $\Gamma' = \{P_1 \cdots P_m, Q_1 \cdots Q_k\}$ *be a set of polynomials over variables* $\{x_1 \cdots x_n\}$ *and* $\{y_1 \cdots y_k\}$ *such that* $Q_1 \cdots Q_k$ *are polynomials of the form* $Q_j = y_j - (a_{j0} + \sum_i a_{ij} x_i)$ *for some constants* $a_{ij} \in \mathbb{F}$. *A Trinomial-*$\Pi\Sigma$*-PC refutation* $R_1 \cdots R_s$ *of* Γ *is a Polynomial Calculus refutation of* Γ'*, such that each* R_ℓ *is either an affine form or a trinomial in* $\{x_1 \cdots x_n\}$ *and* $\{y_1 \cdots y_k\}$.

Trinomial-$\Pi\Sigma$-PC essentially allows each line in the proof to be a $\Sigma\Pi\Sigma$ circuit in X with the top fan-in bounded by 3. We will measure the size of a Trinomial-$\Pi\Sigma$-PC proof by the number of lines, which is clearly polynomially equivalent to the number of monomials in X, Y. This proof system seems quite restricted, especially since it can no longer trivially simulate Polynomial Calculus unlike $\Sigma\Pi\Sigma$-PC. But surprisingly, the Pigeonhole Principle and Tseitin formulas, for which we have lower bounds for Polynomial Calculus, have small proofs in Trinomial-$\Pi\Sigma$-PC.

2.A.1 Proof of the Intersection lemma

Here we prove the Intersection lemma and some of its variants that will be used throughout the rest of the paper.

Lemma 3. *"Substitution Lemma"*

Let $R(z-a_1)\cdots(z-a_k) = 0$ *and* $Rp(z) = 0$ *be two equations in a Depth-*d'*-PC refutation, where* R *is any polynomial and* p *is a univariate polynomial of degree* d *in* z *such that* $p(a_i) \neq 0$ *for any* i. *Then, we can derive the equation* $R = 0$ *in* $O(kd|R|)$ *lines where* $|R|$ *is the number of monomials in* R.

Proof. Consider the base case of $k = 1$. Starting with $R(z-a_1) = 0$, we can successively derive $Rz^i - Ra_1^i = 0$ for $i \in \{2 \cdots d\}$ by multiplying with the appropriate polynomials in z. This takes $O(d|R|)$ lines in total. Then adding these equations up with the appropriate coefficients we obtain $Rp(z) - Rp(a) = 0$. Since $p(a) \neq 0$ and $Rp(z) = 0$, we have $R = 0$. Now, multiplying every line

of the above derivation with $(z-a_2)\cdots(z-a_k)$, we have a derivation of $R(z-a_2)\cdots(z-a_k)=0$ from $R(z-a_1)\cdots(z-a_k)=0$ and $Rp(z)=0$. The lemma now follows by induction over k. □

Lemma 4. *Let $Q(z-a)=0$ and $Q\prod_{i=1}^{k}(z-b_i)=0$ be two equations in Trinomial-ΠΣ-PC, where Q is a monomial and $a \neq b_i$ for any i. Then we can derive $Q=0$ in $O(k)$ lines.*

Proof. The proof is by induction on k. The base case, when $k=0$, is trivial. Assume that the lemma is true for some $k-1 \geq 0$. Let $z_1 = z-a$, $z_2 = z-b_1$ and $Q_1 = \prod_{i=2}^{k}(z-b_i)$. The equations are then represented as

$$Qz_1 = 0 \tag{2.1}$$

$$QQ_1z_2 = 0 \tag{2.2}$$

$$z_1 = z-a \tag{2.3}$$

$$z_2 = z-b \tag{2.4}$$

Multiplying equation (2.1) by Q_1, we have

$$QQ_1z_1 = 0 \tag{2.5}$$

Let $c=a-b$. By subtracting (2.4) from (2.3) we derive

$$z_1 - z_2 + c = 0 \tag{2.6}$$

Now multiplying the above equation by the monomial QQ_1, we derive the trinomial

$$QQ_1z_1 - QQ_1z_2 + cQQ_1 = 0$$

But since we already have $QQ_1z_1 = 0$ from (2.5) and $QQ_1z_2 = 0$ from (2.2), we obtain

$$cQQ_1 = 0$$

Since $c \neq 0$, we derive $QQ_1 = 0$. Therefore, we now have the equations

$$Q(z-a) = 0$$

$$Q\prod_{i=2}^{k}(z-b_i) = 0$$

The proof of the lemma thus follows from the induction hypothesis. Since it only takes a constant number of lines to go from the case of k to the case of $k-1$, the total number of lines in the derivation is $O(k)$.

□

We now generalize this lemma as follows.

Lemma 5. *"Intersection Lemma"*
Let A and B be two sets of constants in $\mathbb{F}$. Let $\prod_{a\in A}(z-a) = 0$ and $\prod_{b\in B}(z-b) = 0$ be two equations in Trinomial-$\Pi\Sigma$-PC. Then there is a proof of $\prod_{c\in A\cap B}(z-c) = 0$ in Trinomial-$\Pi\Sigma$-PC of length $O(|A \setminus B| \cdot |B \setminus A|)$

Proof. We will prove the lemma by induction over the size of $|A \setminus B|$. The base case when $|A \setminus B| = 0$ trivially follows since $A = A \cap B$.

Now for any two sets A and B such that $|A \setminus B| > 0$, let the equations be labeled as follows

$$\prod_{a\in A}(z-a) = 0 \tag{2.7}$$

$$\prod_{b\in B}(z-b) = 0 \tag{2.8}$$

Let $A_0 = A \setminus B$ and $B_0 = B \setminus A$. Choose an element $a_1 \in A_0$. Let $A_1 = A \setminus \{a_1\}$ and $A_2 = A_0 \setminus \{a_1\}$. Let Q_1 be the monomial $\prod_{a \in A_1}(z-a)$ and Q_2 be the monomial $\prod_{a \in A_2}(z-a)$. Then equation (2.7) can be written as

$$Q_1(z-a_1) = 0 \tag{2.9}$$

Multiplying (2.8) by Q_2 we get

$$\prod_{b \in B \cup A_2}(z-b) = 0 \tag{2.10}$$

Note that there are no squared terms in the monomial since A_2 and B are disjoint. The above equation can be rewritten as

$$\prod_{b \in A_1 \cup B_0}(z-b) = 0 \tag{2.11}$$

since $A_1 \cup B_0 = B \cup A_2$. Note that A_1 and B_0 are also disjoint. Hence we can write the above equation as

$$Q_1 \prod_{b \in B_0}(z-b) = 0 \tag{2.12}$$

Now since $a_1 \notin B_0$, we can apply Lemma 4 on equations (2.9) and (2.12) to get

$$Q_1 = 0$$

i.e.

$$\prod_{a \in A_1}(z-a) = 0 \tag{2.13}$$

in $O(|B_0|) = O(|B \setminus A|)$ lines.

Now we have two sets of constants A_1 and B with corresponding equations (2.13) and (2.8) such that $|A_1 \setminus B| = |A \setminus B| - 1$. Thus the lemma follows by induction. The total number of lines is $O(|A \setminus B| \cdot |B \setminus A|)$.

□

Remark It is easy to see that starting with $Q\prod_{a\in A}(z-a) = 0$ and $Q\prod_{b\in B}(z-b) = 0$, we can still apply the Intersection Lemma to get $Q\prod_{c\in A\cap B}(z-c) = 0$ for any monomial Q.

2.A.2 Simulating syntactic CP* in Trinomial-ΠΣ-PC over $\mathbb{Q}$

We are now ready to state and prove Theorem 0, which first appeared in [RT08a].

For each possible derivation rule in a Cutting Planes proof, we will now show how to derive in Trinomial-ΠΣ-PC the translation of the result of applying the rule on a line or a pair of lines, given their translations.

Simulating Addition For the addition rule, given the translations of two lines $\sum_i a_{ij}x_i \geq b_j$ and $\sum_i a_{ik}x_i \geq b_k$ in CP*, we will derive the translation of their sum $\sum_i (a_{ik}+a_{ij})x_i \geq b_j + b_k$. The following lemma suffices.

Lemma 6. *Simulating addition*

Let $x(x-1)\cdots(x-a) = 0$ and $y(y-1)\cdots(y-b) = 0$ be two equations in a Trinomial-ΠΣ-PC refutation with $a \geq b$. Then we can derive

$$(x+y)(x+y-1)\cdots(x+y-(a+b)) = 0$$

using $O(ab)$ lines.

Proof. Let $z = x+y$. We will first derive the range of values z can take when $y = j$, for all $j \in \{0\cdots b\}$. Let $x_i = x - i$ for $i \in \{0\cdots a\}$, $y_j = y - j$ for $j \in \{0\cdots b\}$ and $z_k = z - k$ for

$k \in \{0 \cdots a+b\}$. Also, for $S \subseteq \{0 \cdots b\}$, let $Y_S = \prod_{j \in \{0 \cdots b\} \setminus S} y_j$. We denote $Y_{\{j\}}$ simply by Y_j. Note that $Y_A Y_B = 0$ if $A \cup B = \{0 \cdots b\}$. Then we have

$$z_j = x_0 + y_j$$

Multiplying the above equation by the monomial Y_j, we have

$$z_j Y_j - x_0 Y_j - y_j Y_j = 0$$

Since $y_j Y_j = \prod_{j \in \{0 \cdots b\}} y_j = 0$, we have

$$z_j Y_j - x_0 Y_j = 0 \tag{2.14}$$

It is easy to derive for $i \in \{0 \cdots a\}$

$$z_j - z_{j+i} - i = 0$$

Multiplying the above equation by the monomial Y_j, we have

$$z_j Y_j - z_{j+i} Y_j - i Y_j = 0 \tag{2.15}$$

Subtracting this from (2.14) we get

$$z_{j+i} Y_j - x_0 Y_j + i Y_j = 0 \tag{2.16}$$

By the definition of x_i we have

$$x_i = x_0 - i$$

Multiplying the above equation by the monomial Y_j, we get

$$x_i Y_j - x_0 Y_j + iY_j = 0$$

Subtracting the above equation from (2.16) we get

$$z_{j+1} Y_j - x_i Y_j = 0$$

Thus, for all $i \in \{0 \cdots a\}$ we derive

$$z_{j+i} Y_j - x_i Y_j = 0$$

From the above $a+1$ equations, we can inductively derive for $i \in \{0 \cdots a\}$

$$z_j \cdots z_{j+i} Y_j - x_0 \cdots x_i Y_j = 0$$

as follows. For $i \in \{1 \cdots a\}$, using

$$z_j \cdots z_{j+i-1} Y_j - x_0 \cdots x_{i-1} Y_j = 0$$

we can derive

$$z_j \cdots z_{j+i} Y_j - x_0 \cdots x_{i-1} z_{j+i} Y_j = 0 \tag{2.17}$$

by multiplying with z_{j+1}. Now multiplying

$$z_{j+i} Y_j - x_i Y_j = 0$$

by the monomial $x_0 \cdots x_{i-1}$, we derive

$$x_0 \cdots x_{i-1} z_{j+i} Y_j - x_0 \cdots x_i Y_j = 0 \tag{2.18}$$

Subtracting (2.18) from (2.17) we get

$$z_j \cdots z_{j+i} Y_j - x_0 \cdots x_i Y_j = 0$$

using $O(j)$ monomials. Therefore, we have

$$z_j \cdots z_{j+a} Y_j - x_0 \cdots x_a Y_j = 0 \tag{2.19}$$

and since $x_0 \cdots x_a = 0$, we derive

$$z_j \cdots z_{j+a} Y_j = 0$$

We derive the above for every $j \in \{0 \cdots b\}$ using a total of $O(ab)$ lines. Multiplying the above line by $\{z_k : 0 \leq k < j\} \cup \{z_k : j+a < k \leq a+b\}$, we have for all $j \in \{0 \cdots b\}$

$$z_0 \cdots z_{a+b} Y_j = 0$$

Now note that the set of monomials $\{Y_j : j \in \{0 \cdots b\}\}$ have no common root. Therefore we can apply the Intersection Lemma repeatedly to derive $z_0 \cdots z_{a+b} = 0$ as follows. Starting with

$$z_0 \cdots z_{a+b} Y_{\{0 \cdots j\}} = 0$$

and

$$z_0 \cdots z_{a+b} Y_{j+1} = 0$$

and applying the Intersection Lemma with $A = \{0 \cdots b\} \setminus \{0 \cdots j\}$ and $B = \{0 \cdots b\} \setminus \{j+1\}$ we get

$$z_0 \cdots z_{a+b} Y_{\{0 \cdots j+1\}} = 0$$

using $O(j)$ lines. Thus using $O(b^2)$ lines we get

$$z_0 \cdots z_{a+b} = 0$$

and the total number of lines is $O(ab+b^2)$. □

Corollary 1. *Given the translations of $\sum_i a_{ij}x_i \geq b_j$ and $\sum_i a_{ik}x_i \geq b_k$, we can derive in Trinomial-ΠΣ-PC the translation of $\sum_i (a_{ik}+a_{ij})x_i \geq b_j + b_k$ in $O((\sum_i a_{ij}^+ - b_j)(\sum_i a_{ik}^+ - b_k))$ lines*

Proof. Use the above lemma for $x = \sum_i a_{ij}x_i - b_j$, $a = \sum_i a_{ij}^+ - b_j$ and $y = \sum_i a_{ik}x_i - b_k$, $b = \sum_i a_{ik}^+ - b_k$. □

Simulating multiplication by a constant We use the following lemma to derive the translation of $c\sum_i c_{ij}x_i \geq cd_j$ in Trinomial-ΠΣ-PC from the translation of $\sum_i c_{ij}x_i \geq d_j$

Lemma 7. *Let $(z-a_1)\cdots(z-a_k) = 0$ be an equation in Trinomial-ΠΣ-PC. We can derive the equation*

$$(z'-ca_1)\cdots(z'-ca_k) = 0$$

where $z' = cz$ in Trinomial-ΠΣ-PC for any $c \in \mathbb{Q}$ in $O(k)$ lines.

Proof. The proof is by induction on k. For $k = 0$, the derivation is trivial. Let $z_i = z - a_i$ and $z'_i = z' - ca_i$ for $i \in \{1 \cdots k\}$. Then, for any $k \geq 1$, we are given the equation

$$z_1 \cdots z_k = 0$$

and we want to derive

$$z'_1 \cdots z'_k = 0$$

Since, $z' = cz$, we get $z'_1 = z' - ca_1 = cz_1$ and thus multiplying with $z_2 \cdots z_k$ we get

$$z'_1 z_2 \cdots z_k - cz_1 \cdots z_k = 0$$

But since $z_1 \cdots z_k = 0$ as above, we get

$$z'_1 z_2 \cdots z_k = 0$$

Now by the induction hypothesis we have a derivation of $z'_2 \cdots z'_k = 0$ from $z_2 \cdots z_k = 0$. By multiplying each step of this derivation by z'_1, we have derived $z'_1 \cdots z'_k = 0$ from $z'_1 z_2 \cdots z_k = 0$.

□

Corollary 2. *Given the translation of $\sum_i c_{ij} x_i \geq d_j$, we can derive the translation of $c\sum_i c_{ij} x_i \geq cd_j$ in Trinomial-ΠΣ-PC in $O(\sum_i c_{ij}^+ - d_j)$ lines*

Proof. Use the above lemma for $z = \sum_i c_{ij} x_i - d_j$ and
$(a_1 \ \cdots \ a_k) = (0 \ \cdots \ \sum_i c_{ij}^+ - d_j)$ □

Simulating division by a constant Given the translation of a line $c\sum_i a_{ij} x_i \geq b_j$ in Cutting Planes for some $c > 0$, we will now derive the translation of $\sum_i a_{ij} x_i \geq \lceil b_j / c \rceil$ by the lemma below. We need the following corollary of Lemma 6

Corollary 3. *Let $z = \sum_i a_{ij} x_i$ be an equation in Trinomial-ΠΣ-PC, where x_i are boolean variables. Then we can derive*

$$z\Big(z-1\Big)\cdots\Big(z-\big(\sum_i a_{ij}^+\big)\Big) = 0$$

in $O((\sum_i a_i^+)^2)$ lines.

Proof. Let $a = \sum_{i=1}^{n} a_{ij}^+$ and let $b = \sum_{i=1}^{n/2} a_{ij}^+$. Assume that we have derived the equations

$$z_1\Big(z_1-1\Big)\cdots\Big(z_1-\big(\sum_{i=1}^{n/2} a_i^+\big)\Big) = 0$$

$$z_2\Big(z_2-1\Big)\cdots\Big(z_2-\big(\sum_{i=n/2+1}^{n}a_i^+\big)\Big)=0$$

for $z_1=\sum_{i=1}^{n/2}a_{ij}x_i$ and $z_2=\sum_{i=n/2+1}^{n}a_{ij}x_i$. We can use Lemma 6 on the above two equations to derive the required equation in $O(b(a-b))$ lines. Continuing this recursively for the above two lines, the total number of lines $L(a)$ to derive $z\Big(z-1\Big)\cdots\Big(z-\big(\sum_i a_i^+\big)\Big)=0$ is given by the recurrence $L(a)=L(b)+L(a-b)+O(b(a-b))$, which gives $L(a)=O(a^2)$ by an easy induction. □

Lemma 8. *Simulating Division by a constant*
Let $(cz-b)(cz-(b+1))\cdots(cz-d)=0$ *be an equation in Trinomial-*$\Pi\Sigma$*-PC where* $z=\sum_i a_{ij}x_i$ *such that* x_i *are boolean variables,* $b<d$ *and* $c>0$*. We can derive*

$$(z-\lceil b/c\rceil)(z-(\lceil b/c\rceil+1))\cdots(z-\lfloor d/c\rfloor)=0$$

using $O((\sum_i a_i^+)^2+(\sum_i a_i^+)(d-b))$ *lines.*

Proof. Using Corollary 3 we can derive the following equation in $O((\sum_i a_i^+)^2)$ lines.

$$z\Big(z-1\Big)\cdots\Big(z-\big(\sum_i a_{ij}^+\big)\Big)=0 \tag{2.20}$$

Now, using Lemma 7 on the equation $(cz-b)(cz-(b+1))\cdots(cz-d)=0$ with the multiplication constant equal to $1/c$, we can derive

$$z(z-b/c)\cdots(z-d/c)=0 \tag{2.21}$$

Note that the constants in parentheses in the above equation are *rational*, and the smallest integer that appears is $\lceil b/c\rceil$ and the largest integer that appears is $\lfloor d/c\rfloor$. Using the Intersection Lemma with equations (2.20) and (2.21), we see that only the integer values are retained from (2.21) which gives us

$$(z - \lceil b/c \rceil)(z - (\lceil b/c \rceil + 1)) \cdots (z - \lfloor d/c \rfloor)$$

using $O((\sum_i a_i^+)(d-b))$ lines.

□

Corollary 4. *Given the translation of a line $c\sum_i a_{ij}x_i \geq b_j$ for some $c > 0$, we can derive in Trinomial-ΠΣ-PC the translation of $\sum_i a_{ij}x_i \geq \lceil b_j/c \rceil$ in $O(c(\sum_i a_{ij}^+)^2)$lines*

Proof. Apply the above lemma for $z = \sum_i a_{ij}x_i$. □

This completes the simulation of a syntactic CP* proof in Trinomial-ΠΣ-PC with the simulation having size polynomial in n and the coefficient size of the original proof.

2.A.3 Simulating semantic CP* in Trinomial-ΠΣ-PC over $\mathbb{Q}$

In this section we extend the above simulation to include semantic CP*, hence completing the proof of Theorem 1. Let $L_1 \equiv \sum_i a_i x_i \geq d_1$, $L_2 \equiv \sum_i b_i x_i \geq d_2$ be two lines in a Cutting Planes proof and let $L_3 \equiv \sum_i c_i x_i \geq d_3$ be a semantic consequence of L_1 and L_2. Let $y = \sum_i a_i x_i$, $z = \sum_i b_i x_i$ and $w = \sum_i c_i x_i$. Let $A = \{0 \cdots \sum_i a_i^+\}$, $B = \{0 \cdots \sum_i b_i^+\}$ and $C = \{0 \cdots \sum_i c_i^+\}$. Using Lemma 3, we can derive the equations

$$\prod_{a \in A}(y - a) = 0$$

$$\prod_{b \in B}(z - b) = 0$$

$$\prod_{c \in C}(w - c) = 0$$

This restricts the values that can be taken by the tuple (y, z, w) to the three dimensional grid $A \times B \times C$. Let a point (i, j, k) in the grid be *infeasible* if the tuple (y, z, w) never evaluates to

it for any assignment to $\{x_i\}$. Our first step is to derive *infeasibility equations* of the form

$$\prod_{\substack{a\in A\\ a\neq i}}(y-a)\prod_{\substack{b\in B\\ b\neq j}}(z-b)\prod_{\substack{c\in C\\ c\neq k}}(w-c)=0$$

which for $(i,j,k)\in A\times B\times C$ tells us that the point (i,j,k) in the grid is infeasible for the tuple (y,z,w).

Lemma 9. *For every infeasible point $(i,j,k)\in A\times B\times C$, an infeasibility equation of the above form can be derived in $O((\sum_i a_i^+)^2(\sum_i b_i^+)^2(\sum_i c_i^+)^2)$ lines*

Proof. We proceed by induction on n. Let $y_\ell=\sum_{i=1}^{\ell}a_ix_i$ and z_ℓ, w_ℓ, A_ℓ, B_ℓ, C_ℓ be defined analogously. For the base case of $n=1$, the equations defining the grid are $y_1(y_1-a_1)=0$, $z_1(z_1-b_1)=0$ and $w_1(w_1-c_1)=0$. The only feasible points in the grid are $(0,0,0)$ and (a_1,b_1,c_1), and thus for every other tuple we will derive an infeasibility equation. We show the derivation for one such tuple $(a_1,0,0)$. Starting with

$$y_1=a_1x_1$$

$$z_1=b_1x_1$$

derive

$$z_1-b_1=b_1(x_1-1)$$

and multiply by y_1 to derive

$$y_1(z_1-b_1)=a_1b_1x_1(x_1-1)=0$$

Multiplying the above equation by (w_1-c_1), we have our required infeasibility equation.

To continue the induction and derive all possible infeasibility equations, we observe that a point (i,j,k) for (y_ℓ,z_ℓ,w_ℓ) is infeasible if and only if the points (i,j,k) and $(i-a_\ell,j-b_\ell,k-c_\ell)$

are infeasible for $(y_{\ell-1}, z_{\ell-1}, w_{\ell-1})$. Therefore, assuming the latter, we derive the former as follows. Given

$$\prod_{\substack{a\in A_{\ell-1}\\ a\neq i}} (y_{\ell-1}-a) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j}} (z_{\ell-1}-b) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k}} (w_{\ell-1}-c) = 0$$

and

$$\prod_{\substack{a\in A_{\ell-1}\\ a\neq i-a_\ell}} (y_{\ell-1}-a) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j-b_\ell}} (z_{\ell-1}-b) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k-c_\ell}} (w_{\ell-1}-c) = 0$$

we will derive

$$\prod_{\substack{a\in A_{\ell}\\ a\neq i}} (y_{\ell}-a) \prod_{\substack{b\in B_{\ell}\\ b\neq j}} (z_{\ell}-b) \prod_{\substack{c\in C_{\ell}\\ c\neq k}} (w_{\ell}-c) = 0$$

Starting with the equations

$$y_\ell = y_{\ell-1} + a_\ell x_\ell$$

$$z_\ell = z_{\ell-1} + b_\ell x_\ell$$

$$w_\ell = w_{\ell-1} + c_\ell x_\ell$$

multiply each by $(x_\ell - 1)$ to derive

$$y_\ell(x_\ell - 1) = y_{\ell-1}(x_\ell - 1)$$

$$z_\ell(x_\ell - 1) = z_{\ell-1}(x_\ell - 1)$$

$$w_\ell(x_\ell - 1) = w_{\ell-1}(x_\ell - 1)$$

From the above equations, it is easy to derive (see Lemma 6)

$$(x_\ell - 1) \prod_{\substack{a \in A_{\ell-1} \\ a \neq i}} (y_\ell - a) \prod_{\substack{b \in B_{\ell-1} \\ b \neq j}} (z_\ell - b) \prod_{\substack{c \in C_{\ell-1} \\ c \neq k}} (w_\ell - c) \tag{2.22}$$

$$= (x_\ell - 1) \prod_{\substack{a \in A_{\ell-1} \\ a \neq i}} (y_{\ell-1} - a) \prod_{\substack{b \in B_{\ell-1} \\ b \neq j}} (z_{\ell-1} - b) \prod_{\substack{c \in C_{\ell-1} \\ c \neq k}} (w_{\ell-1} - c) \tag{2.23}$$

$$= 0 \tag{2.24}$$

Similarly, we derive from the three starting equations

$$y_\ell - a_\ell = y_{\ell-1} + a_\ell(x_\ell - 1)$$

$$z_\ell - b_\ell = z_{\ell-1} + b_\ell(x_\ell - 1)$$

$$w_\ell - c_\ell = w_{\ell-1} + c_\ell(x_\ell - 1)$$

Multiplying by x_ℓ we have

$$(y_\ell - a_\ell)x_\ell = y_{\ell-1}x_\ell$$

$$(z_\ell - b_\ell)x_\ell = z_{\ell-1}x_\ell$$

$$(w_\ell - c_\ell)x_\ell = w_{\ell-1}x_\ell$$

Analogous to the above we can derive

$$x_\ell \prod_{\substack{a\in A_{\ell-1}\\ a\neq i-a_\ell}} (y_\ell-(a+a_\ell)) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j-b_\ell}} (z_\ell-(b+b_\ell)) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k-c_\ell}} (w_\ell-(c+c_\ell)) \tag{2.25}$$

$$= x_\ell \prod_{\substack{a\in A_{\ell-1}\\ a\neq i-a_\ell}} (y_{\ell-1}-a) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j-b_\ell}} (z_{\ell-1}-b) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k-c_\ell}} (w_{\ell-1}-c) \tag{2.26}$$

$$= 0 \tag{2.27}$$

As $A_{\ell-1} \cup \{a+a_\ell : a \in A_{\ell-1}\} \subseteq A_\ell$ (similarly for B_ℓ and C_ℓ), we have from equations (2.22) and (2.25)

$$(x_\ell-1) \prod_{\substack{a\in A_\ell\\ a\neq i}} (y_\ell-a) \prod_{\substack{b\in B_\ell\\ b\neq j}} (z_\ell-b) \prod_{\substack{c\in C_\ell\\ c\neq k}} (w_\ell-c) = 0 \tag{2.28}$$

$$x_\ell \prod_{\substack{a\in A_\ell\\ a\neq i}} (y_\ell-a) \prod_{\substack{b\in B_\ell\\ b\neq j}} (z_\ell-b) \prod_{\substack{c\in C_\ell\\ c\neq k}} (w_\ell-c) = 0 \tag{2.29}$$

Adding the above two equations, we derive the required one. □

The next step is to use the ranges of y and z specified in lines L_1 and L_2 to narrow down the possible values that can be taken by w. Our goal will be to get an equation of the form

$$\prod_{c\in C'} (w-c) = 0$$

such that each c in C' is feasible for w under the constraints L_1 and L_2 on y and z respectively.

Let P_i be the translation of L_i in Trinomial-ΠΣ-PC, for $i = 1,2,3$. Let $I_{a,b}$ denote the set

of all infeasibility equations for points of the form (a,b,k) for some $k \in C$. For an equation P of the form $\prod_{a\in A_1}(y-a)\prod_{b\in B_1}(z-a)\prod_{c\in C_1}(w-a)=0$, denote by $\mathcal{R}_y(P)$ the set A_1, that is the range of values specified by the equation for the variable y. $\mathcal{R}_z$ and $\mathcal{R}_w$ are defined analogously. We describe how to obtain the set C' by the algorithm w-FEASIBLE which operates on the range sets.

```
1 C' ← ∅
2 for (a,b) ∈ R_y(P1) × R_z(P2) do
3 |   S ← C for I ∈ I_{a,b} do
4 |   |   S ← S ∩ R_w(I)
5 |   end
6 |   C' ← C' ∪ S
7 end
8 return C'
```

Algorithm 2: w-feasible(P_1,P_2)

Consider a pair $(a,b) \in \mathcal{R}_y(P_1) \times \mathcal{R}_z(P_2)$. For any equation $I \in I_{a,b}$, $\mathcal{R}_w(I)$ gives a list of possible values the variable w can take when $(y,z)=(a,b)$. By Lemma 9, $(y,z,w)=(a,b,c)$ is infeasible if and only if there is an equation $I \in I_{a,b}$ such that $c \notin \mathcal{R}_w(I)$. Therefore, $\bigcap\limits_{I\in I_{a,b}} \mathcal{R}_w(I)$ is precisely the feasible set of values for w, given $(y,z)=(a,b)$. C' is the union of such sets over all possible pairs $(a,b) \in \mathcal{R}_y(P_1) \times \mathcal{R}_z(P_2)$ and hence is the set of all feasible values of w.

This algorithm over range sets can be easily translated to a proof of $\prod_{c\in C'}(w-c)=0$ from P_1 and P_2 in Trinomial-ΠΣ-PC as follows. To simulate the inner **for** loop, we use the Intersection lemma inductively over all equations in $I_{a,b}$ to get equations $J_{a,b}$ such that $\mathcal{R}_w(J_{a,b}) = \bigcap\limits_{I\in I_{a,b}} \mathcal{R}_w(I)$. Note that $\mathcal{R}_y(J_{a,b}) = A \setminus \{a\}$ and $\mathcal{R}_z(J_{a,b}) = B \setminus \{b\}$. Thus using the Intersection lemma again inductively over the set $\{J_{a,b}\}$ analogous to Lemma 6 would give an equation free of y and z, where w ranges over $\bigcup\limits_{(a,b)} \mathcal{R}_w(J_{a,b})$. Any semantic consequence P_3 must be such that $\mathcal{R}_w(P_3) \supseteq C'$ and hence is easily derived.

2.A.4 Simulating syntactic CP* in Depth-5-PC over $\mathbb{F}_{p^m}$

We will now carry out the simulation in Section 2.*A*.2 in Depth-d-PC over a large enough field extension F_{p^m} of a finite field F_p. This will be of use in the next section, where we simulate $AC^0[p]$-Frege in Depth-d-PC over F_{p^m}. For the following discussion, we set $d = 5$

To represent large integers over $\mathbb{F}_{p^m}$, we choose a primitive element α and for a boolean x_i perform the linear transformation $y_i = 1 + (\alpha - 1)x_i$. Since x_i is boolean, y_i is essentially equivalent to the mapping $x_i \mapsto \alpha^{x_i}$. The expression $\sum_i a_i x_i$ is thus represented as $\alpha^{\sum_i a_i x_i}$. We will show that all the steps of the simulation in section 2.A.2 can still be performed after this transformation.

Theorem 12. *Depth-d-PC over F_{p^m} can simulate syntactic Cutting Planes with the number of lines polynomial in n and the coefficient size, where m is logarithmic in n and the coefficient size.*

Let s_1 be the coefficient size of the Cutting Planes proof. Define $s = ns_1$. Choose m to be the smallest integer such that $2s^2 < p^m - 1$. Let α be an arbitrary primitive element of $\mathbb{F}_{p^m}$.

Definition 25. *Translation of Cutting Planes to Depth-d-PC over $\mathbb{F}_{p^m}$*

Given a line $\sum_i a_i x_i \geq b_i$ in Cutting Planes, the translation of the above line is defined as the following lines, where y_i and y are new variables.

$$y_i = (\alpha^{a_i} - 1)x_i + 1$$

$$y = \prod_i y_i$$

$$(y - \alpha^{b_i})(y - \alpha^{b_i+1}) \cdots (y - \alpha^{\sum_i a_i^+}) = 0$$

An integer c such that $0 \leq c \leq s$ is represented as α^c, whereas for $-s \leq c < 0$ we represent it as $\alpha^{-|c|} \equiv \alpha^{(p^m-1)-|c|}$. Since $2s \leq 2s^2 < p^m - 1$, these representations are unique.

The following lemmas will be largely similar to the ones in the previous section.

Simulating Addition To simulate the addition rule, it suffices to show the following

Lemma 10. *Let A and B be two sets of constants in any field and let $C = \{ab \mid a \in A, b \in B\}$. Let $\prod_{a\in A}(x-a) = 0$ and $\prod_{b\in B}(x-b) = 0$ be two equations in Depth-d-PC. Let $z = xy$. Then the equation*

$$\prod_{c\in C}(z-c) = 0$$

can be derived in $O(|A||B|)$ lines.

Proof. Let $A = \{a_i\}, B = \{b_i\}$, $x_i = x - a_i$ and $y_i = y - b_i$. Note that $x_1 \cdots x_{|A|} = 0 = y_1 \cdots y_{|B|}$. Let $X_j = \prod_{i\neq j} x_i$. Starting with

$$z = xy$$

we can derive

$$z = (x-a_j)y + a_j y$$

Now multiplying the above equation by X_j, we have

$$zX_j = x_1 \cdots x_{|A|} y + a_j y X_j = a_j y X_j$$

Subtracting $a_j b_i X_j$ on both sides we can derive for every i the equation

$$(z - a_j b_i)X_j = a_j(y - b_i)X_j$$

Now, similar to Lemma 6, we can derive from the $|B|$ equations above the equation

$$(z-a_j b_1)\cdots(z-a_j b_{|B|})X_j = a_j y_1 \cdots y_{|B|} X_j = 0$$

Thus for every j we have the equation

$$(z-a_j b_1)\cdots(z-a_j b_{|B|})X_j = 0$$

Multiplying each of the above $|A|$ equations with the missing terms, we can obtain for every j,

$$\prod_{c \in C}(z-c)X_j = 0$$

Using the Intersection Lemma inductively as in Lemma 6, we obtain the required equation.

□

Corollary 5. *Given the translations of $\sum_i a_{ij}x_i \geq b_j$ and $\sum_i a_{ik}x_i \geq b_k$ in Depth-d-PC over F_{p^m}, we can derive the translation of $\sum_i (a_{ik}+a_{ij})x_i \geq b_j + b_k$ in $O((\sum_i a_{ij} - b_j)(\sum_i a_{ik} - b_k))$ lines*

Proof. Use the above lemma for $y_1 = \prod_i((\alpha^{a_{ij}} - 1)x_i + 1)$, $y_2 = \prod_i((\alpha^{a_{ik}} - 1)x_i + 1)$, $A = \{\alpha^{b_j}, \alpha^{b_j+1} \cdots \alpha^{\sum_i a^+_{ij}}\}$, $B = \{\alpha^{b_k}, \alpha^{b_k+1} \cdots \alpha^{\sum_i a^+_{ik}}\}$ □

Simulating Multiplication

Lemma 11. *Let A be a set of constants in any field and let c be a positive integer. Let $A^c = \{a^c \mid a \in A\}$. Let $\prod_{a \in A}(x-a) = 0$ be an equation in the Depth-d-PC. Then we can derive the equation*

$$\prod_{a \in A^c}(x^c - a) = 0$$

in $O(|A|)$ lines.

Proof. Let $x_i = x - a_i$ and $x'_i = x_i^c$. Then the given equation becomes $x_1 \cdots x_{|A|} = 0$, and we want to derive $x'_1 \cdots x'_{|A|} = 0$. The proof is by induction on $|A|$. If $|A| = 0$ then we have nothing to prove. Assume that the statement is true for $|A| \leq k-1$ for some $k \geq 1$. Consider an expression of the form $\prod_{a \in A}(x-a) = 0$, where $|A| = k$. If $|A^c| < k$, then clearly there exists a set $A_1 \subset A$ such that $A_1^c = A^c$, and the required equation follows from the induction hypothesis. If $|A^c| = k$, from the given equation, it is easy to derive

$$xx_2 \cdots x_k - a_1 x_2 \cdots x_k = 0$$

Multiplying the above equation with x, we have

$$x^2 x_2 \cdots x_k - a_1 x x_2 \cdots x_k = 0$$

Adding a_1 times the former equation to the latter, we have

$$x^2 x_2 \cdots x_k - a_1^2 x_2 \cdots x_k = 0$$

Proceeding in a similar way, we can derive

$$x^c x_2 \cdots x_k - a_1^c x_2 \cdots x_k = 0$$

or equivalently

$$x_1' x_2 \cdots x_k = 0$$

Now by the induction hypothesis, we have a proof of $x_2' \cdots x_k' = 0$ from $x_2 \cdots x_k = 0$. Multiplying each line of the proof by x_1' we arrive at a proof of the required equation. □

Corollary 6. *Given the translation of $\sum_i a_{ij} x_i \geq b_j$ in Depth-d-PC over F_{p^m} and an integer $c < p^m - 1$, we can derive the translation of $\sum_i c a_{ij} x_i \geq c b_j$ in $O((\sum_i a_{ij} - b_j))$ lines*

Proof. Use the above lemma for $y = \prod_i ((\alpha^{a_{ij}} - 1)x_i + 1)$, $A = \{\alpha^{b_j}, \alpha^{b_j+1} \cdots \alpha^{\sum_i a_{ij}^+}\}$ □

Note that previous two lemmas hold over any field. For the following lemma, we will use the fact that we are working over F_{p^m} where $s^2 < p^m - 1$.

Simulating Division The proof of the following corollary is analogous to Corollary 3.

Corollary 7. *Let $x = \prod_i((\alpha^{b_{ij}} - 1)x_i + 1)$ be a variable where x_i are boolean. We can derive*

$$(x-1)(x-\alpha)\cdots(x-\alpha^{\sum_i b_{ij}^+}) = 0$$

in $O((\sum_i b_{ij}^+)^2)$ lines

Lemma 12. *Let $(x^c - \alpha^{ca_1})\cdots(x^c - \alpha^{ca_k}) = 0$ be an equation in Depth-d-PC over $\mathbb{F}_{p^m}$, where a_i are distinct and x is of the form $\prod_i((\alpha^{b_{ij}} - 1)x_i + 1)$ where x_i are boolean. There is a proof of the equation*

$$(x-\alpha^{a_1})\cdots(x-\alpha^{a_k}) = 0$$

in $O((\sum_i a_i^+)^2)$ lines

Proof. Using Corollary 7, we can derive

$$(x-1)(x-\alpha)\cdots(x-\alpha^{\sum_i b_{ij}^+}) = 0 \tag{2.30}$$

in $O((\sum_i b_{ij}^+)^2)$ lines. Since $\sum_i |b_{ij}| < s$, any term $(x-\alpha^b)$ that appears in the above equation is such that $b \in [0,s]$ or $b \in [p^m - 1 - s, p^m - 2]$.

The proof is by induction on k. Consider the case of $k = 1$, when we have the equation $x^c - \alpha^{ca_1} = 0$ where $a_1 \leq s$ without loss of generality. If $c \nmid p^m - 1$, then it has a unique root α^{a_1}. If $c \mid p^m - 1$, then the roots are of the form $\alpha^{a_i + j(p^m-1)/c}$ for $j \in \{0\cdots c-1\}$. But since $2s^2 < p^m - 1$,

$$c \leq s < (p^m - 1)/2s \leq (p^m - 1)/2c \tag{2.31}$$

Therefore any root α^b such that $b \neq a_1$ is such that $b \geq a_i + (p^m - 1)/c > s$. Also, we

have

$$\begin{aligned} b &\leq a_i + (p^m - 1)(c-1)/c \\ &= p^m - 1 - ((p^m-1)/c - a_i) \\ &< p^m - 1 - ((p^m-1)/c - s) \\ &< p^m - 1 - s \end{aligned}$$

where the last inequality is due to (2.31). Therefore the only root α^b to the equation $x^c - \alpha^{ca_1} = 0$ such that $b \in [0,s]$ or $b \in [p^m - 1 - s, p^m - 2]$ is α^{a_1}. Starting with the equation $x^c - \alpha^{ca_1} = 0$ it is easy to derive

$$(x - \alpha^{a_1})Q(x) = 0 \tag{2.32}$$

where $Q(x) = x^{c-1} + \alpha x^{c-2} + \cdots + \alpha^{c-1}$, just by expanding the above equation into its monomials. Now by our discussion above, for any term $(x - \alpha^b)$ that appears in the equation (2.30), $Q(\alpha^b) \neq 0$. Therefore, using the Substitution lemma with equations (2.30) and (2.32) we derive $x - \alpha^{a_1} = 0$ if this term appears in (2.30), else we derive $1 = 0$. Therefore, this gives a derivation of $x - \alpha^{a_1} = 0$ from the equation $x^c - \alpha^{ca_1} = 0$.

For the induction step, by multiplying every step in the above derivation with $(x^c - \alpha^{ca_2}) \cdots (x^c - \alpha^{ca_k})$, we obtain a derivation of

$$(x - \alpha^{a_1})(x^c - \alpha^{ca_2}) \cdots (x^c - \alpha^{ca_k}) = 0$$

from

$$(x^c - \alpha^{ca_1}) \cdots (x^c - \alpha^{ca_k}) = 0$$

The lemma now follows by induction.

□

Corollary 8. *Given the translation of $c\sum_i a_{ij}x_i \geq b_j$ in Depth-d-PC over F_{p^m} for an integer $c < p^m - 1$, we can derive the translation of $\sum_i a_{ij}x_i \geq \lceil b_j/c \rceil$ in $O((c\sum_i a_{ij}^+)^2)$ lines*

Proof. Let the equation

$$(y^c - \alpha^{b_j}) \cdots (y^c - \alpha^{c\sum_i a_{ij}^+}) = 0 \tag{2.33}$$

be obtained from the translation of $c\sum_i a_{ij}x_i \geq b_j$, where $y = \prod_i((\alpha^{a_{ij}} - 1)x_i + 1)$. We first use Corollary 7 to derive

$$(y-1)(y-\alpha)\cdots(y-\alpha^{\sum_i a_{ij}^+}) = 0$$

in $(\sum_i a_{ij}^+)^2$ lines. Using Lemma 11 on the above equation, we get

$$(y^c - 1)(y^c - \alpha^c)\cdots(y^c - \alpha^{c\sum_i a_{ij}^+}) = 0 \tag{2.34}$$

in $\sum_i a_{ij}^+$ lines. Using the Intersection Lemma on equations (2.33) and (2.34), we get

$$(y^c - \alpha^{c\lceil b_j/c \rceil}) \cdots (y^c - \alpha^{c\sum_i a_{ij}^+}) = 0$$

We now use the previous lemma to derive

$$(y - \alpha^{\lceil b_j/c \rceil}) \cdots (y - \alpha^{\sum_i a_{ij}^+}) = 0$$

which is the required equation.

□

This completes the proof of Theorem 12

2.B Simulating $\mathsf{AC}^0[q]$-Frege in Depth-7-PC over $\mathbb{F}_{p^m}$

2.B.1 Case of $q = p$

For the purpose of this section, we set $d = 7$. We will use the simulation of $\mathsf{AC}^0[p]$-Frege in [MP98] to show that the same can be carried out in Depth-d-PC over $\mathbb{F}_{p^m}$. We fix m to be a large enough integer such that $m = O(\mathsf{poly}(\log(n)))$, so that the field we are working over is quasipolynomially sized. Below we describe the proof system of [MP98] and their simulation of $\mathsf{AC}^0[p]$-Frege.

The Proof System of Maciel and Pitassi

Maciel and Pitassi [MP98] define a proof system with mod p, negation, AND, OR and threshold connectives, based on the system PTK by Buss and Clote [BC96] which we describe below.

Connectives Let $x_1 \cdots x_n$ be boolean variables. For $0 \leq j < p$, let $\oplus^p_j(x_1 \cdots x_n)$ denote the connective which is 1 if and only if $\sum_i x_i = j \mod p$. For any integer t, let $Th_t(x_1 \cdots x_n)$ denote the connective which is 1 if and only if $\sum_i x_i \geq t$. Let $\wedge(x_1 \cdots x_n)$, $\vee(x_1 \cdots x_n)$ denote AND and OR connectives of arity n and $\neg$ denote the NOT gate.

Formulas A *formula* is recursively defined as follows. Input variables $x_1 \cdots x_n$ are formulas of size 1 and depth 1. A formula φ is an expression of the form $g(\varphi_1 \cdots \varphi_k)$, where g is any of the connectives described above and $\varphi_1 \cdots \varphi_k$ are formulas. The $depth(\varphi)$ is defined as $\sum_{i=1}^k depth(\varphi_i) + 1$. The $size(\varphi)$ is defined as $\sum_{i=1}^k size(\varphi_i) + k + 1$ if g is not a threshold connective, and it is defined as $\sum_{i=1}^k size(\varphi_i) + t + k + 1$ if g is a threshold connective of the form $Th_t(\varphi_1 \cdots \varphi_k)$.

Cedents and Sequents A cedent Γ is defined as a sequence of formulas $\varphi_1 \cdots \varphi_k$. We will use capital Greek letters to denote cedents. A sequent is an expression of the form $\Gamma \to \Delta$, where Γ and Δ are cedents. The interpretation of a sequent is that the AND of all the formulas in Γ implies the OR of all the formulas in Δ. The size and depth of a cedent are respectively the sum of sizes and the maximum of depths of all the formulas in it. The size of a sequent is the sum of sizes of both cedents, and the depth is the maximum of the depths of both cedents.

Definition of a Proof A proof in this system is defined as a sequence of sequents $\mathcal{S}_1 \cdots \mathcal{S}_m$ such that each $\mathcal{S}_i$ is either an initial sequent, or is derived from sequents $\mathcal{S}_j$ for $j < i$ through one of the rules listed below. The size and depth of a proof are respectively the sum of sizes and the maximum of depths of all sequents in it.

The initial sequents and the derivation rules are listed below.

The proof system of Maciel and Pitassi [MP98]

initial sequents

1. $\varphi \to \varphi$ for any formula φ
2. $\to \wedge() \; ; \vee() \to$
3. $\oplus^p_j() \to$ for $1 \leq j < p \; ; \to \oplus^p_0()$
4. $Th_t() \to$
5. $\to Th_0(\varphi_1 \cdots \varphi_k)$ for any $k \geq 0$

structural rules

weakening: $\dfrac{\Gamma, \Delta \to \Gamma'}{\Gamma, \varphi, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \Delta'}{\Gamma \to \Gamma', \varphi, \Delta'}$

contract: $\dfrac{\Gamma, \varphi, \varphi, \Delta \to \Gamma'}{\Gamma, \varphi, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \varphi, \varphi, \Delta'}{\Gamma \to \Gamma', \varphi, \Delta'}$

permute: $\dfrac{\Gamma, \varphi_1, \varphi_2, \Delta \to \Gamma'}{\Gamma, \varphi_2, \varphi_1, \Delta \to \Gamma'} \quad \dfrac{\Gamma \to \Gamma', \varphi_1, \varphi_2, \Delta'}{\Gamma \to \Gamma', \varphi_2, \varphi_1, \Delta'}$

cut rule

$$\frac{\Gamma, \varphi \to \Delta \qquad \Gamma' \to \varphi, \Delta'}{\Gamma, \Gamma' \to \Delta, \Delta'}$$

logical rules

$\neg$: $\dfrac{\Gamma \to \varphi, \Delta}{\neg\varphi, \Gamma \to \Delta} \quad \dfrac{\varphi, \Gamma \to \Delta}{\Gamma \to \neg\varphi, \Delta}$

$\wedge$-left: $\dfrac{\varphi_1, \wedge(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{\wedge(\varphi_1 \cdots \varphi_k), \Gamma \to \Delta}$

$\wedge$-right: $\dfrac{\Gamma \to \varphi_1, \Delta \qquad \Gamma \to \wedge(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \wedge(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

$\vee$-left: $\dfrac{\varphi_1, \Gamma \to \Delta \qquad \vee(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{\vee(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

$\vee$-right: $\dfrac{\Gamma \to \varphi_1, \vee(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \vee(\varphi_1 \cdots \varphi_k), \Delta}$

$\oplus_i$-left: $\dfrac{\varphi_1, \oplus^p_{i-1}(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta \qquad \oplus^p_i(\varphi_2 \cdots \varphi_k), \Gamma \to \varphi_1, \Delta}{\oplus^p_i(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

$\oplus_i$-right: $\dfrac{\varphi_1, \Gamma \to \oplus^p_{i-1}(\varphi_2 \cdots \varphi_k), \Delta \qquad \Gamma \to \varphi_1, \oplus^p_i(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to \oplus^p_i(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

Th_t-left: $\dfrac{Th_t(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta \qquad \varphi_1, Th_{t-1}(\varphi_2 \cdots \varphi_k), \Gamma \to \Delta}{Th_t(\varphi_1, \varphi_2 \cdots \varphi_k), \Gamma \to \Delta}$

Th_t-right: $\dfrac{\Gamma \to \varphi_1, Th_t(\varphi_2 \cdots \varphi_k), \Delta \qquad \Gamma \to Th_{t-1}(\varphi_2 \cdots \varphi_k), \Delta}{\Gamma \to Th_t(\varphi_1, \varphi_2 \cdots \varphi_k), \Delta}$

Translating lines

We will now define translations of lines in the above proof system. For a formula φ, we denote its translation in Depth-d-PC by $tr(\varphi)$. Let $x_1 \cdots x_n$ be the variables of the original proof. Below we list the translations for a formula built with each connective. The interpretation is that for any formula φ, $tr(\varphi) = 0$ if and only if φ is true.

$$tr(x_i) = 1 - x_i$$

$$tr(\vee(\varphi_1 \cdots \varphi_k)) = \prod_i (tr(\varphi_i))$$

$$tr(\wedge(\varphi_1 \cdots \varphi_k)) = 1 - \prod_i tr(\neg\varphi_i)$$

$tr(\oplus_i^p(\varphi_1 \cdots \varphi_k)) = (\sum_{j=1}^k \varphi_j - i)^{p-1}$ for $0 \leq i < p$

$tr(Th_t(\varphi_1 \cdots \varphi_k)) = (y - \alpha^t) \cdots (y - \alpha^k)$
where $y = \prod_i((\alpha - 1)tr(\neg\varphi_i) + 1)$

$tr(\neg\varphi) = 1 - tr(\varphi)$ if φ does not contain a Th_t connective

$tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = (y - 1) \cdots (y - \alpha^{t-1})$
where $y = \prod_i((\alpha - 1)tr(\neg\varphi_i) + 1)$, for $t \geq 1$

The translation $tr(\mathcal{S})$ of a sequent $\mathcal{S}$ of the form $\varphi_1 \cdots \varphi_k \rightarrow \varphi'_1 \cdots \varphi'_m$ is given by the equation

$$\prod_{i=1}^{k} tr(\neg\varphi_i) \prod_{j=1}^{m} tr(\varphi'_j) = 0$$

Note that the translations of all the connectives except the threshold connective take only boolean values over $\mathbb{F}_{p^m}$.

Simulating proofs

We now describe the connection between $\mathsf{AC}^0[p]$-Frege and the proof system of Maciel and Pitassi. By the following theorem of Allender [All89], any $\mathsf{AC}^0[p]$ circuit can converted to a depth three circuit of a special form.

Theorem 13. ***[All89]***
Any $\mathsf{AC}^0[p]$ circuit can be converted to a quasipolynomial sized depth three circuit with an unweighted Threshold gate at the top, MOD_p gates of quasipolynomial fan-in in the middle and $\wedge$ gates of polylogarithmic fan-in at the bottom

Depth three circuits with an unweighted Threshold, $\wedge$ or $\vee$ gate at the top, MOD_p gates in the middle and $\wedge$ gates of polylogarithmic fan-in in the size of the circuit at the bottom are referred to as *flat circuits* by [MP98]. For an $\mathsf{AC}^0[p]$ circuit φ, its *flattening* $fl(\varphi)$ is defined

as the flat circuit given by the above theorem. Proofs in $\mathsf{AC}^0[p]$-Frege can be thought of as a list of sequents such that every formula that appears in each of them is an $\mathsf{AC}^0[p]$ circuit. For a sequent $\varphi_1 \cdots \varphi_k \to \varphi'_1 \cdots \varphi'_m$ that appears in a $\mathsf{AC}^0[p]$-Frege proof, we can define a flattening of the sequent $fl(\varphi_1) \cdots fl(\varphi_k) \to fl(\varphi'_1) \cdots fl(\varphi'_m)$ in the proof system of Maciel and Pitassi. A *flat proof* of such a sequent is such that every formula that appears in the proof is a flat circuit. The simulation theorem of [MP98] states the following

Theorem 14. ***[MP98]***

Let $\mathcal{S}$ be a sequent which has a depth d proof in $\mathsf{AC}^0[p]$-Frege. Then its flattening $fl(\mathcal{S})$ has a flat proof of size $2^{(\log n)^{O(d)}}$ in the proof system of Maciel and Pitassi.

We will show that flat proofs can be simulated in Depth-d-PC by showing the following

Theorem 15. *Let $\mathcal{S}$ be a sequent which has a flat proof of size s in the proof system of Maciel and Pitassi. Then there is a proof of the equation $tr(\mathcal{S})$ in Depth-d-PC from the equations $x_i(x_i-1)=0$ with $poly(s)$ lines.*

To prove the above theorem, it is sufficient to show that for each rule that derives a sequent $\mathcal{S}_3$ from sequents $\mathcal{S}_1$ and $\mathcal{S}_2$, there is a derivation of the equation $tr(\mathcal{S}_3)$ from the equations $tr(\mathcal{S}_1)$, $tr(\mathcal{S}_2)$ and $x_i(x_i-1)=0$ in Depth-d-PC. Below we show how each such rule can be simulated.

Simulating Initial sequents

Here we will show how to derive translations of the initial sequents from $x_i(1-x_i)=0$.

Lemma 13. *Let φ be any formula of depth three which only contains the $\oplus_i^p$, $\neg$, $\wedge$ and $\vee$ connectives. Then the equation $tr(\varphi)(1-tr(\varphi))=0$ can be derived from $x_i(x_i-1)=0$ in Depth-d-PC*

Proof. Easily follows from repeated application of Lemmas 6, 10 and 11 at each level. □

Lemma 14. *The translation of the initial sequent $\varphi \to \varphi$ can be derived from $x_i(x_i-1)=0$ in Depth-d-PC for any flat circuit φ*

Proof. If φ is a flat circuit without Threshold gates, this follows by Lemma 13 since the translation of the sequent $\varphi \to \varphi$ is simply $tr(\varphi)(1-tr(\varphi)) = 0$. If φ contains a top Threshold gate, the translation of the given sequent states that a variable y such that $y = \prod_{i=1}^{k}((\alpha-1)tr(\neg\varphi_i)+1)$ satisfies $(y-1)\cdots(y-\alpha^k) = 0$, where φ_i are formulas without Threshold gates. Thus we can derive $tr(\neg\varphi_i)(1-tr(\neg\varphi_i)) = 0$ as in Lemma 13 and then use Lemma 7 to derive $(y-1)\cdots(y-\alpha^k) = 0$. □

The initial sequents 2,3 and 4 are dummies and do not require translating. The initial sequent 5 can be derived using Lemma 7 since in a flat proof each of the inputs to the Threshold connective do not contain Threshold connectives.

Simulating structural rules

The simulation of the weakening rule just involves multiplying the given equation by the translation of the new formula φ that appears. The permutation rule is trivial since the translation of a sequent is invariant under application of the permutation rule. To simulate the contraction rule, we need to show that for every formula φ, we can derive from $(tr(\varphi))^2 = 0$ the equation $tr(\varphi) = 0$. When φ is a formula which does not involve a Threshold connective, this is can be done by using Lemma 13. When φ is a flat circuit with a Threshold gate at the top, the following lemma suffices.

Lemma 15. *Let $(y-\alpha^{a_1})^2\cdots(y-\alpha^{a_m})^2 = 0$ be an equation in Depth-d-PC where a_i are distinct integers less than p^m-1 and $y = \prod_{i=1}^{k}((\alpha-1)tr(\neg\varphi_i)+1)$ such that φ_i are flat formulas with no Threshold gates. The equation $(y-\alpha^{a_1})\cdots(y-\alpha^{a_m}) = 0$ can be derived in $O(\max(m,k^2))$ lines.*

Proof. The proof is by induction on m. The case of $m = 0$ is trivial. Using Lemma 7 we can derive the range of values of the variable y, i.e. an equation of the form

$$(y-1)\cdots(y-\alpha^k) = 0 \tag{2.35}$$

Let $Q = (y-\alpha^{a_1})(y-\alpha^{a_2})^2\cdots(y-\alpha^{a_m})^2$ and $Q_1 = (y-\alpha^{a_2})^2\cdots(y-\alpha^{a_m})^2$. Then the given equation can be written as

$$Q(y-\alpha^{a_1}) = 0 \tag{2.36}$$

Multiplying equation (2.35) with Q if it does not contain the term $(y-\alpha^{a_1})$, else multiplying it with Q_1, we arrive at

$$Q \prod_{1\leq i\leq k\,,\, i\neq a_1} (y-\alpha^i)$$

Using Lemma 4 with equations (2.35) and (2.36), we get $Q = 0$. The lemma now follows by induction since assuming there is a derivation of $(y-\alpha^{a_2})\cdots(y-\alpha^{a_m}) = 0$ from $(y-\alpha^{a_2})^2\cdots(y-\alpha^{a_m})^2$, this derivation can be multiplied by $(y-\alpha^{a_1}) = 0$ to get the required equation from $Q = 0$. □

Simulating the cut rule

Let $Q = tr(\neg\Gamma)tr(\Delta)$ and $Q' = tr(\neg\Gamma')tr(\Delta')$. Let $y = tr(\varphi)$ if φ does not contain Threshold gates, else let $y = \prod_{i=1}^{k}((\alpha-1)tr(\neg\varphi_i)+1)$ where $\varphi = Th_t(\varphi_1\cdots\varphi_k)$. Then the cut rule can be translated to the following statement

Lemma 16. *Given the equations $Q(y-a_1)\cdots(y-a_k) = 0$ and $Q'(y-b_1)\cdots(y-b_m) = 0$ where $a_1\cdots a_k$ and $b_1\cdots b_m$ are disjoint sets of constants from the field, derive $QQ' = 0$*

Proof. Multiply the first equation by Q' and the second equation by Q, and use the contraction rule to make sure the resulting equations are square free. Then required equation now follows easily from the Intersection Lemma. □

Simulating $\wedge$,$\vee$, $\oplus_i^p$ and $\neg$ rules

The rules for $\neg$, $\wedge$-left and $\vee$-right are trivially simulated since the translation remains invariant.

For the $\wedge$-right and $\vee$-left, the simulation reduces to the following lemma, where $Q = tr(\neg\Gamma)tr(\Delta)$.

Lemma 17. *Given the equations $Qy_1 = 0$ and $Qy = 0$ where y_1 and y take boolean values, derive the equation $Qyy_1 = 0$*

Proof. Follows from Lemma 7 □

For the $\wedge$-right rule, the above lemma can be instantiated with $y_1 = tr(\varphi_1)$ and $y = tr(\wedge(\varphi_2 \cdots \varphi_k))$. Since $\wedge(\varphi_1 \cdots \varphi_k)$ is being derived, each of the formulas φ_i must be free of Threshold gates. Thus the fact that y and y_1 are boolean is easily derived from Lemma 13. A similar simulation works for the $\vee$-left rule.

The simulation for $\oplus_i^p$ gates is analogous to the above. Let $Q = tr(\neg\Gamma)tr(\Delta)$, and $x_i = tr(\varphi_i)$. The $\oplus_1^p$-left rule then translates to the following lemma. The simulations for the other $\oplus_i^p$ rules are similar.

Lemma 18. *Given the equations*

$$x_1(1 - z_2^{p-1}) = 0$$

and

$$(1 - x_1)(1 - (1 - z_2)^{p-1}) = 0$$

derive $(1 - (1 - z_1)^{p-1}) = 0$, where $z_1 = x_1 + \cdots x_n$, $z_2 = x_2 + \cdots x_n$ and x_i are boolean variables.

Proof. Starting with the equation

$$z_1 = x_1 + z_2$$

Multiply by $(1 - x_1)$ on both sides and subtract $(1 - x_1)$ to get

$$(z_1 - 1)(1 - x_1) = x_1(1 - x_1) + (z_2 - 1)(1 - x_1) = (z_2 - 1)(1 - x_1)$$

Now, we can raise both sides of the equation to the exponent $p - 1$, and use the fact that $(1 - x_1)^{p-1} = (1 - x_1)$ (which is easily derived using Lemma 11) to get

$$(z_1-1)^{p-1}(1-x_1) = (z_2-1)^{p-1}(1-x_1)$$

But since from the second equation of our hypothesis, $(z_2-1)^{p-1}(1-x_1) = (1-x_1)$ and thus

$$(1-(z_1-1)^{p-1})(1-x_1) \tag{2.37}$$

Now consider the equation

$$z_1 - 1 = x - 1 + z_2$$

obtained by subtracting one from $z_1 = x_1 + z_2$

Multiplying by x on both sides, we get

$$(z_1-1)x_1 = x(x-1) + z_2x = z_2x$$

Again, raising to the exponent $p-1$ and noting that $x_1^{p-1} = x_1$ and $z_2^{p-1}x_1 = x_1$ we have

$$(z_1-1)^{p-1}x_1 = z_2^{p-1}x_1 = x_1$$

and thus

$$(1-(z_1-1)^{p-1})x_1 = 0$$

Adding equation (2.37) to the above we get the required equation

□

Simulating Th_t rules

Let $Q = tr(\neg\Gamma)tr(\Delta)$, and $x_i = tr(\neg\varphi_i)$. The Th_t-left rule translates to the following lemma.

The case of Th_t-right is similar.

Lemma 19. *Given the equations*

$$(z_2 - 1)\cdots(z_2 - \alpha^{t+1}) = 0$$

and

$$x_1(z_2 - 1)\cdots(z_2 - \alpha^{t}) = 0$$

derive

$$(z_1 - 1)\cdots(z_1 - \alpha^{t+1}) = 0$$

where $z_1 = \prod_{i=1}^{k}((\alpha-1)x_i + 1)$, $z_2 = \prod_{i=2}^{k}((\alpha-1)x_i + 1)$ *and* x_i *are boolean variables.*

Proof. It is easy to derive the equation

$$z_1 = (\alpha x_1 + 1 - x_1)z_2$$

Multiplying the above equation with $(1 - x_1)$ we get

$$z_1(1 - x_1) = (1 - x_1)^2 z_2 = (1 - x_1)z_2$$

since x_1 is boolean. Subtracting $\alpha^i(1 - x_1)$ on both sides we get

$$(z_1 - \alpha^i)(1 - x_1) = (z_2 - \alpha^i)(1 - x_1)$$

for every i in $\{0\cdots t+1\}$. From these $t+1$ equations it is easy to derive (see Lemma 6)

$$(z_1 - 1)\cdots(z_1 - \alpha^{t+1})(1 - x_1) = (z_2 - 1)\cdots(z_2 - \alpha^{t+1})(1 - x_1) = 0 \tag{2.38}$$

Multiplying the equation $z_1 = (\alpha x_1 + 1 - x_1)z_2$ with x_1 we get

$$z_1 x_1 = \alpha x_1^2 z_2 = \alpha x_1 z_2$$

Again, subtracting $\alpha^{i+1} x_1$ we get

$$(z_1 - \alpha^{i+1}) x_1 = (z_2 - \alpha^i) x_1$$

for every i in $\{0 \cdots t\}$. Once again, we combine them to derive

$$(z_1 - \alpha) \cdots (z_1 - \alpha^{t+1}) x_1 = (z_2 - 1) \cdots (z_2 - \alpha^t) x_1 = 0$$

Multiplying the above equation with $z_1 - 1$ and adding it to equation (2.38), we get the required equation. □

This completes the simulation of flat proofs in Depth-d-PC.

2.B.2 Case of $q \neq p$

We now extend the simulation of the previous section to show that $\mathsf{AC}^0[q]$-Frege can be simulated in Depth-d-PC over F_{p^m}, for distinct primes p and q, hence proving Theorem 2. Using the theorem of Maciel and Pitassi (Theorem 14 above) for $\mathsf{AC}^0[q]$-Frege, we obtain a flat proof with $\oplus_i^q$ connectives. To simulate it, we can reuse the lemmas of the previous section, except for the $\oplus_i^q$ connectives. To define their translation, choose m such that $q \mid p^m - 1$ and let $r = (p^m - 1)/q$. The translation is now defined as

$$tr(\oplus_i^q(\varphi_1 \cdots \varphi_k)) = \left((y - \alpha^{ir})\right)^{p^m - 1}$$

where $y = \prod_i ((\alpha^r - 1) tr(\neg \varphi_i) + 1)$ and $tr(\neg \oplus_i^q (\varphi_1 \cdots \varphi_k)) = 1 - tr(\oplus_i^q (\varphi_1 \cdots \varphi_k))$

The proof of the main lemma below simulating one of the rules is quite similar to the one

in the previous section.

Lemma 20. *Given the equations*

$$x_1(1-(y_2-1)^{p^m-1})=0$$

and

$$(1-x_1)(1-(y_2-\alpha^r)^{p^m-1})=0$$

derive

$$(1-(y_1-\alpha^r)^{p^m-1})=0$$

where $y_1=\prod_{i=1}^{k}((\alpha^r-1)x_i+1)$, $y_2=\prod_{i=2}^{k}((\alpha^r-1)x_i+1)$ *and* x_i *are boolean variables*

Proof. It is easy to derive

$$y_1=(\alpha^r x_1+1-x_1)y_2$$

Multiplying the above equation with x_1 we have

$$y_1x_1=\alpha^r y_2x_1^2=\alpha^r y_2x_1$$

since x_1 is boolean. By subtracting $\alpha^r x_1$ we can now derive

$$(y_1-\alpha^r)x_1=\alpha^r x_1(y_2-1)$$

Raising the above equation to the power p^m-1, we get

$$(y_1-\alpha^r)^{p^m-1}x_1=x_1(y_2-1)^{p^m-1}$$

since x_1 is boolean. Subtracting the above equation from x_1, we get

$$(1-(y_1-\alpha^r)^{p^m-1})x_1 = (1-(y_2-1)^{p^m-1})x_1 = 0 \tag{2.39}$$

By multiplying with $1-x_1$ we can derive from $y_1 = (\alpha^r x_1 + 1 - x_1)y_2$ the equation

$$y_1(x_1-1) = y_2(x_1-1)$$

Carrying out a derivation similar to the above, we get

$$(1-(y_1-\alpha^r)^{p^m-1})(x_1-1) = (1-(y_2-\alpha^r)^{p^m-1})(x_1-1) = 0 \tag{2.40}$$

Adding equations (2.39) and (2.40) we get the required equation. □

2.C Simulating TC^0-Frege in Depth-d-PC over $\mathbb{F}_{p^m}$

In this section, we show that a TC^0-Frege proof of depth d_0 can be transformed into a Depth-d-PC proof over $\mathbb{F}_{p^m}$, where $d = O(d_0)$, proving Theorem 3. In the previous section we translated $Th_t(\varphi_1 \cdots \varphi_k)$ as

$$tr(Th_t(\varphi_1 \cdots \varphi_k)) = (y-\alpha^t)\cdots(y-\alpha^k)$$

$$tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = (y-1)\cdots(y-\alpha^{t-1})$$

where $y = \prod_i((\alpha-1)tr(\neg\varphi_i)+1)$. Clearly this translation requires $tr(\varphi_i)$ to be boolean and can itself take non-boolean values. Since there is only one top Threshold gate in a flat circuit, the formulae φ_i were threshold free and thus $tr(\varphi_i)$ only took on boolean values. But in a TC^0-Frege proof, the formulae φ_i can themselves contain Threshold gates and thus $tr(\varphi_i)$ may be non-boolean. To fix this problem, we redefine the translation of a Threshold gate to be the following, essentially forcing it to be boolean.

$$tr(Th_t(\varphi_1 \cdots \varphi_k)) = \big((y-\alpha^t)\cdots(y-\alpha^k)\big)^{p^m-1}$$

where $y = \prod_i((\alpha-1)tr(\neg\varphi_i)+1)$ and $tr(\neg Th_t(\varphi_1 \cdots \varphi_k)) = 1 - tr(Th_t(\varphi_1 \cdots \varphi_k))$.

It is easy to generalize Lemma 13 to derive the fact that the above translation only takes boolean values. Now, note that any rule other than the Th_t is unaffected by this new translation since it only assumes that its arguments are boolean and hence we can use the lemmas of the previous section directly. However, simulation of the Th_t rule relies on the old translation. To bridge the gap, we only need to show that the old and new translations of Th_t and $\neg Th_t$ are interchangeable within the proof system.

Lemma 21. *Given the equation*

$$\big((y-\alpha^t)\cdots(y-\alpha^k)\big)^{p^m-1} = 0$$

we can derive

$$(y-\alpha^t)\cdots(y-\alpha^k) = 0$$

and vice versa.

Proof. In the forward direction, the required equation is easily derived by repeated application of the contraction rule. The other direction is trivial. □

Lemma 22. *Given the equation*

$$1 - \big((y-\alpha^t)\cdots(y-\alpha^k)\big)^{p^m-1} = 0$$

we can derive

$$(y-1)\cdots(y-\alpha^{t-1}) = 0$$

and vice versa.

Proof. In the forward direction, since y is a Threshold gate with k arguments, we can derive

$$(y-1)\cdots(y-\alpha^k)=0$$

and thus

$$((y-1)\cdots(y-\alpha^k))^{p^m-1}=0$$

But since we have $\big((y-\alpha^t)\cdots(y-\alpha^k)\big)^{p^m-1}=1$ from the given equation, we get

$$\big((y-1)\cdots(y-\alpha^{t-1})\big)^{p^m-1}=0$$

Using the contraction rule repeatedly gives the required equation.

In the reverse direction, Let $y_1=\big((y-\alpha^t)\cdots(y-\alpha^k)\big)^{p^m-1}$. Then as mentioned earlier, we can derive using Lemma 13

$$y_1(1-y_1)=0$$

Using the contraction rule on the above equation, we get

$$(y-\alpha^t)\cdots(y-\alpha^k)(1-y_1)=0 \tag{2.41}$$

Multiplying the given equation $(y-1)\cdots(y-\alpha^{t-1})=0$ by $(1-y_1)$ and using the Intersection Lemma with equation (2.41), we get $1-y_1=0$, which is the required equation. □

2.D Dealing with large coefficients

2.D.1 Properties of addition

In this section we derive some basic properties of addition.

The following lemma shows that our system can prove the associativity of $\oplus$.

Lemma 23. *For bits y, z, w, let* $H(y,z) := y \wedge z$ *and let* $H(y,z,w) := (y \wedge z) \vee (z \wedge w) \vee (w \wedge y)$ *which is one if and only if* $y+z+w \geq 2$. $H(.)$ *denotes the carry bit generated by adding together up to three bits. The following are easily proved since they involve only a constant number of variables.*

$$\vdash H(y,z,w) - H(y,z \oplus w) \oplus H(z,w) \tag{2.42}$$

$$z_1 + w_1 - (z_2 + w_2) \vdash H(y,z_1,w_1) - H(y,z_2,w_2) \tag{2.43}$$

$$\vdash H(H(y,z \oplus w), H(z,w)) \tag{2.44}$$

If c_i *are carry bits in* $\mathbf{y} \oplus \mathbf{z}$*, then*

$$\vdash c_{i+1} - H(y_i, z_i, c_i) \tag{2.45}$$

For bits a,b,c,d,e*,*

$$\vdash H(a,b,c) + H(a \oplus b \oplus c, d, e) - H(a,b,d) - H(a \oplus b \oplus d, c, e) \tag{2.46}$$

Lemma 24. *For any three bit vectors* $\mathbf{y}$*,* $\mathbf{z}$ *and* $\mathbf{w}$

$$\vdash (\mathbf{y} \oplus \mathbf{z}) \oplus \mathbf{w} - \mathbf{y} \oplus (\mathbf{z} \oplus \mathbf{w})$$

Proof. Let $\mathbf{y}_{left} := (\mathbf{y} \oplus \mathbf{z}) \oplus \mathbf{w}$ and $\mathbf{y}_{right} := \mathbf{y} \oplus (\mathbf{z} \oplus \mathbf{w})$. Let $d_i^{\mathbf{y},\mathbf{z}}$ be the carry bit to the i^{th} position in $\mathbf{y} \oplus \mathbf{z}$. Let $d_i^{\mathbf{w}}$ be the carry bit to the i^{th} position in $(\mathbf{y} \oplus \mathbf{z}) \oplus \mathbf{w}$. Similarly define $d_i^{\mathbf{z},\mathbf{w}}$ and $d_i^{\mathbf{y}}$. We will derive inductively for every i

$$\vdash d_i^{\mathbf{y},\mathbf{z}} + d_i^{\mathbf{w}} - (d_i^{\mathbf{z},\mathbf{w}} + d_i^{\mathbf{y}}) \tag{2.47}$$

$$\vdash \mathbf{y}_{left}(i) - \mathbf{y}_{right}(i)$$

This is easily derived for $i = 1$. Suppose for some $i \geq 1$ the above lines have been derived. By (2.45) of Lemma 23, we derive

$$\vdash d_{i+1}^{\mathbf{y},\mathbf{z}} - H(\mathbf{y}(i), \mathbf{z}(i), d_i^{\mathbf{y},\mathbf{z}})$$

$$\vdash d_{i+1}^{\mathbf{w}} - H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus d_i^{\mathbf{y},\mathbf{z}}, \mathbf{w}(i), d_i^{\mathbf{w}})$$

since $\mathbf{y} \oplus \mathbf{z}(i) = \mathbf{y}(i) \oplus \mathbf{z}(i) \oplus d_i^{\mathbf{y},\mathbf{z}}$. Adding these lines we get

$$\vdash d_{i+1}^{\mathbf{y},\mathbf{z}} + d_{i+1}^{\mathbf{w}} - \big(H(\mathbf{y}(i), \mathbf{z}(i), d_i^{\mathbf{y},\mathbf{z}}) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus d_i^{\mathbf{y},\mathbf{z}}, \mathbf{w}(i), d_i^{\mathbf{w}})\big)$$

Using (2.46) of Lemma 23, we make the derivation

$$\vdash H(\mathbf{y}(i), \mathbf{z}(i), d_i^{\mathbf{y},\mathbf{z}}) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus d_i^{\mathbf{y},\mathbf{z}}, \mathbf{w}(i), d_i^{\mathbf{w}}) \\ - \big(H(\mathbf{y}(i), \mathbf{z}(i), \mathbf{w}(i)) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{y},\mathbf{z}}, d_i^{\mathbf{w}})\big)$$

Adding this to the line above, we get

$$\vdash d_{i+1}^{\mathbf{y},\mathbf{z}} + d_{i+1}^{\mathbf{w}} - \big(H(\mathbf{y}(i), \mathbf{z}(i), \mathbf{w}(i)) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{y},\mathbf{z}}, d_i^{\mathbf{w}})\big)$$

In a similar fashion, we make the derivation

$$\vdash d_{i+1}^{\mathbf{z},\mathbf{w}} + d_{i+1}^{\mathbf{y}} - \big(H(\mathbf{y}(i), \mathbf{z}(i), \mathbf{w}(i)) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{z},\mathbf{w}}, d_i^{\mathbf{y}})\big)$$

Now, using our induction hypothesis (2.47) and (2.43) of Lemma 23, we derive

$$\vdash H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{y},\mathbf{z}}, d_i^{\mathbf{w}}) - H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{z},\mathbf{w}}, d_i^{\mathbf{y}})$$

The derivation

$$\vdash d_{i+1}^{\mathbf{y},\mathbf{z}} + d_{i+1}^{\mathbf{w}} - (d_{i+1}^{\mathbf{z},\mathbf{w}} + d_{i+1}^{\mathbf{y}})$$

is now easily obtained from the three previous lines.

To derive $\mathbf{y}_{left}(i+1) = \mathbf{y}_{right}(i+1)$, we first make the following derivation

$$d_{i+1}^{\mathbf{y},\mathbf{z}} + d_{i+1}^{\mathbf{w}} - (d_{i+1}^{\mathbf{z},\mathbf{w}} + d_{i+1}^{\mathbf{y}}) \vdash d_{i+1}^{\mathbf{y},\mathbf{z}} \oplus d_{i+1}^{\mathbf{w}} - (d_{i+1}^{\mathbf{z},\mathbf{w}} \oplus d_{i+1}^{\mathbf{y}})$$

since this involves only a constant number of boolean variables. Now, by definition, $\mathbf{y}_{left}(i+1) = \mathbf{y}(i+1) \oplus \mathbf{z}(i+1) \oplus \mathbf{w}(i+1) \oplus d_{i+1}^{\mathbf{y},\mathbf{z}} \oplus d_{i+1}^{\mathbf{w}}$ and by the above two lines this is equal to $\mathbf{y}(i+1) \oplus \mathbf{z}(i+1) \oplus \mathbf{w}(i+1) \oplus d_{i+1}^{\mathbf{z},\mathbf{w}} \oplus d_{i+1}^{\mathbf{y}}$, which is equal to $\mathbf{y}_{right}(i+1)$.

□

The following lemmas show that the addition operations $\mathcal{S}$ and $\oplus$ can be used interchangeably.

Lemma 25. *For $i \leq n$,*

$$\vdash \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^o - \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_i)$$

$$\vdash \mathcal{S}^e(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^e - \mathcal{S}^e(\mathbf{y}_1 \cdots \mathbf{y}_i)$$

Proof. We are going to prove the statement block wise. For odd j, let $w_j = \sum_{k=1}^{i-1} [L_j(\mathbf{y}_k^o)]$. Note that the pair of blocks $(j, j+1)$ in $\mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_i)$ only depend on the corresponding pair of blocks in $\mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1})$ and $L_j(\mathbf{y}_i^o)$. Therefore, restricted to the blocks $(j, j+1)$, the statement of the lemma just depends on w_j and $L_j(\mathbf{y}_i^o)$. Since w_j only takes on n^2 values and $L_j(\mathbf{y}_i^o)$ only takes on n values, there is a polynomial sized proof by completeness. □

Lemma 26. $\vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_i) - \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i$

Proof. Since $\mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) = \mathcal{S}^e(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1})$ and $\mathbf{y}_i = \mathbf{y}_i^e \oplus \mathbf{y}_i^o$ by definition, we have

$$\vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i - \mathcal{S}^e(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^e \oplus \mathbf{y}_i^o$$

From Lemma 24, we have

$$\vdash \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^e \oplus \mathbf{y}_i^o - (\mathbf{y}_i^e \oplus \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^o)$$

Combining the above two derivations, we have

$$\vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i - \mathcal{S}^e(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^e \oplus \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^o$$

Now, using the previous lemma, we are done.

□

The following corollary easily follows from repeated application of the above lemma.

Corollary 9. *For $j < i$, $\vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_i) - \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_j) \oplus \mathcal{S}(\mathbf{y}_{j+1} \cdots \mathbf{y}_i)$.*

Lemma 27. *For every t*

$$\vdash \mathcal{S}(\mathbf{y}_1 X_1 \cdots \mathbf{y}_t X_t) \oplus \mathcal{S}(\mathbf{z}_1 X_1 \cdots \mathbf{z}_t X_t) - \mathcal{S}((\mathbf{y}_1 \oplus \mathbf{z}_1) X_1 \cdots (\mathbf{y}_t \oplus \mathbf{z}_t) X_t)$$

Proof. Assume by induction that we have made the above derivation until $t = i - 1$. Then we have

$$\vdash_1 \mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_iX_i)\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_iX_i)-\mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_{i-1}X_{i-1})\oplus\mathbf{y}_iX_i\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_{i-1}X_{i-1})\oplus\mathbf{z}_iX_i$$

$$\vdash_2 \mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_iX_i)\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_iX_i)-\mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_{i-1}X_{i-1})\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_{i-1}X_{i-1})\oplus\mathbf{y}_iX_i\oplus\mathbf{z}_iX_i$$

$$\vdash_3 \mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_iX_i)\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_iX_i)-\mathcal{S}((\mathbf{y}_1\oplus\mathbf{z}_1)X_1\cdots(\mathbf{y}_{i-1}\oplus\mathbf{z}_{i-1})X_{i-1})\oplus(\mathbf{y}_i\oplus\mathbf{z}_i)X_i$$

$$\vdash_4 \mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_iX_i)\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_iX_i)-\mathcal{S}((\mathbf{y}_1\oplus\mathbf{z}_1)X_1\cdots(\mathbf{y}_t\oplus\mathbf{z}_t)X_t)$$

where $\vdash_1$ and $\vdash_4$ follow by Lemma 26, $\vdash_2$ follows by Lemma 24 and $\vdash_3$ follows by the induction hypothesis. □

Finally, we show how to derive the representation of the sum of two polynomials.

Lemma 28. *Let P and Q be two polynomials. Then $\mathcal{R}(P+Q)=\mathcal{R}(P)\oplus\mathcal{R}(Q)$.*

Proof. Let $X_1\cdots X_t$ be monomials that occur in both P and Q, such that $P=a_1X_1+\cdots+a_tX_t+P_1$ and $Q=b_1X_1+\cdots+b_tX_t+Q_1$. Then from the definition of $\mathcal{R}$ and Corollary 9 we have

$$\vdash \mathcal{R}(P)-\mathcal{S}(\mathbf{a}_1X_1\cdots\mathbf{a}_tX_t)\oplus\mathcal{R}(P_1)$$

$$\vdash \mathcal{R}(Q)-\mathcal{S}(\mathbf{b}_1X_1\cdots\mathbf{b}_tX_t)\oplus\mathcal{R}(Q_1)$$

Using the above, we now have

$$\vdash \mathcal{R}(P)\oplus\mathcal{R}(Q)-\mathcal{S}(\mathbf{a}_1X_1\cdots\mathbf{a}_tX_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{S}(\mathbf{b}_1X_1\cdots\mathbf{b}_tX_t)\oplus\mathcal{R}(Q_1)$$

$$\vdash_1 \mathcal{R}(P)\oplus\mathcal{R}(Q)-\mathcal{S}(\mathbf{a}_1X_1\cdots\mathbf{a}_tX_t)\oplus\mathcal{S}(\mathbf{b}_1X_1\cdots\mathbf{b}_tX_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{R}(Q_1)$$

$$\vdash_2 \mathcal{R}(P)\oplus\mathcal{R}(Q)-\mathcal{S}((\mathbf{a}_1\oplus\mathbf{b}_1)X_1\cdots(\mathbf{a}_t\oplus\mathbf{b}_t)X_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{R}(Q_1)$$

$$\vdash_3 \mathcal{R}(P)\oplus\mathcal{R}(Q)-\mathcal{R}(P+Q)$$

where $\vdash_1$ is by Lemma 24 and $\vdash_2$ is by the previous lemma. $\vdash_3$ is by Corollary 9 and the definition of $\mathcal{R}$.

□

Lemma 29. *For two vectors* $\mathbf{y}$ *and* $\mathbf{z}$, $-(\mathbf{y} \oplus \mathbf{z}) = (-\mathbf{y}) \oplus (-\mathbf{z})$.

Proof. Let $\mathbf{w} = \mathbf{y} \oplus \mathbf{z}$ and let $\mathbf{y_1}$, $\mathbf{z_1}$ be vectors obtained by flipping the bits of $\mathbf{y}$, $\mathbf{z}$ respectively. Let $\mathbf{w_1} = \mathbf{y_1} \oplus \mathbf{z_1}$. It is easy to derive for every i,

$$\vdash \mathbf{y}(i) \oplus \mathbf{z}(i) - \mathbf{y_1}(i) \oplus \mathbf{z_1}(i) \tag{2.48}$$

For $j < m$, let $b_j = \Big(\wedge_{i<j} (\mathbf{y}(i) \oplus \mathbf{z}(i)) \Big) \wedge \neg(\mathbf{y}(j) \oplus \mathbf{z}(j))$ and $b_m = \wedge_{i \leq m} (\mathbf{y}(i) \oplus \mathbf{z}(i))$ be a boolean variable indicating the least index i_0 such that $\mathbf{y}(i_0) \oplus \mathbf{z}(i_0) = 0$. Let c_i be the carry bits in $\mathbf{y} \oplus \mathbf{z}$. We translate boolean formulas into polynomials using the operator $tr()$ defined in Section 2.B.1. We first derive for every j and $i \leq j$,

$$\vdash tr(b_j \rightarrow (c_i = 0))$$

This is done by noting that $c_1 = 0$ and by (2.45) of Lemma 23, $c_i = H(\mathbf{y}(i-1), \mathbf{z}(i-1), c_{i-1})$ for $i > 1$. Assuming by induction that we have derived for some $j > i \geq 1$

$$\vdash tr(b_j \rightarrow (c_i = 0))$$

it is easy to derive

$$\vdash tr(b_j \rightarrow \mathbf{y}(i) \oplus \mathbf{z}(i))$$

Now using the above two derivations with the identity (2.42) of Lemma 23 and the observation $\vdash \mathbf{y}(i) \oplus \mathbf{z}(i) \rightarrow \neg H(\mathbf{y}(i), \mathbf{z}(i))$, we have

$$\vdash tr(c_{i+1} - H(c_i, \mathbf{y}(i) \oplus \mathbf{z}(i)) \oplus H(\mathbf{y}(i), \mathbf{z}(i)))$$

$$\vdash tr(b_j \rightarrow (c_{i+1} = 0))$$

Since $\mathbf{w}(i) = \mathbf{y}(i) \oplus \mathbf{z}(i) \oplus c_i$, for every j and $i \leq j$, we have the derivation

$$\vdash tr(b_j \rightarrow (\mathbf{w}(i) = \mathbf{w_1}(i)))$$

We now want to inductively derive for every j and $i > j$

$$\vdash tr(b_j \rightarrow (\mathbf{w}(i) = \mathbf{w}_1(i) \oplus 1)) \tag{2.49}$$

Let c'_i indicate the carry bits in $\mathbf{y}_1 \oplus \mathbf{z}_1$. Due to the derivation (2.48), we only need to derive for every $i > j$

$$\vdash tr(b_j \rightarrow c_i \oplus c'_i)$$

If $\mathbf{y}(i-1) \oplus \mathbf{z}(i-1) = 0$ (this includes the base case of $i = j+1$), it is easy to derive the following identity independent of the values of c_{i-1} and c'_{i-1}.

$$\vdash tr((\mathbf{y}(i-1) \oplus \mathbf{z}(i-1) = 0) \rightarrow c_i \oplus c'_i)$$

Assuming now that we have derived $\vdash c_i \oplus c'_i$ for some $i > j$, for the case where $\mathbf{y}(i-1) \oplus \mathbf{z}(i-1) = 1$, it is easy to derive

$$\vdash tr((c_i \oplus c'_i) \wedge (\mathbf{y}(i) \oplus \mathbf{z}(i) = 0) \rightarrow c_{i+1} \oplus c'_{i+1})$$

Now consider the vector $\mathbf{w_1} \oplus \mathbf{1}$. By the definition of b_j we have the derivation for all

$i < j$

$$\vdash tr(b_j \rightarrow (\mathbf{w}(i) = 1))$$

and

$$\vdash tr(b_j \rightarrow (\mathbf{w}(j) = 0))$$

Thus it is easy to derive for $i \leq j$

$$\vdash tr(b_j \rightarrow (\mathbf{w} \oplus \mathbf{1}(i) = \mathbf{w}(i) \oplus 1))$$

and for $i > j$

$$\vdash tr(b_j \rightarrow (\mathbf{w} \oplus \mathbf{1}(i) = \mathbf{w}(i)))$$

Combining the above two derivations with (2.49), we have for all i and j

$$\vdash tr(b_j \rightarrow (\mathbf{w} \oplus \mathbf{1}(i) = \mathbf{w}(i) \oplus 1))$$

Since b_j are mutually exclusive, we can eliminate them using techniques similar to Lemma 5 and obtain

$$\vdash \mathbf{w} \oplus \mathbf{1}(i) - \mathbf{w}(i) \oplus 1$$

Hence $\mathbf{w_1} \oplus \mathbf{1}$ the vector obtained by flipping all the bits of $\mathbf{w}$. Therefore, using the definition of $-\mathbf{w}$ and Lemma 24

$$\vdash (-\mathbf{w}) - \mathbf{y_1} \oplus \mathbf{z_1} \oplus \mathbf{1} \oplus \mathbf{1}$$

$$\vdash (-\mathbf{w}) - (-\mathbf{y}) \oplus (-\mathbf{z})$$

□

Lemma 30. *For any vector* $\mathbf{y}$ *of length* $\ell < m-1$*,*

$$\mathbf{y}(m)-1 \vdash (-\mathbf{y})(m)$$

Proof. Since $\mathbf{y}$ is of length ℓ, we have for $\ell < j \leq m$

$$\mathbf{y}(m)-1 \vdash \mathbf{y}(j)-1$$

Let $\mathbf{y}_1$ be the vector obtained by flipping the bits of $\mathbf{y}$. Then we have the derivation for $\ell < j \leq m$

$$\mathbf{y}(m)-1 \vdash \mathbf{y}_1(j)$$

Now, using the identity (2.45) of Lemma 23, we have for $\ell+1 < j \leq m$

$$\mathbf{y}(m)-1 \vdash (\mathbf{y}_1 \oplus \mathbf{1})(j)$$

Since $-\mathbf{y} = \mathbf{y}_1 \oplus \mathbf{1}$, the lemma follows.

□

Lemma 31. *Let* P *be a polynomial represented by a vector* $\mathbf{y}$*. Then* $\vdash \mathcal{R}(-P)-(-\mathbf{y})$*.*

Proof. Let $P = a_1X_1 + \cdots + a_tX_t$. We derive the above by induction on t. Let $P_i = a_1X_1 + \cdots + a_tX_i$ for $i < t$. Then since by Lemma 26, $\vdash \mathcal{R}(P)-(\mathcal{R}(P_{t-1}) \oplus \mathbf{a}_tX_t)$, we have by Lemma 29

$$\vdash (-\mathcal{R}(P))-(-\mathcal{R}(P_{t-1})) \oplus (-\mathbf{a}_tX_t)$$

The lemma now follows from the induction hypothesis and Lemma 26.

□

2.D.2 Non-negative vectors are closed under addition

In this section we show that non-negative vectors of bounded length are closed under the addition $\oplus$. This will be used to show that the vector representations of all the lines of the simulation are bounded in length. Note that some of these claims need not be provable in our proof system.

We first show that given two vectors $\mathbf{y}$ and $\mathbf{z}$ of length ℓ, $\mathbf{y} \oplus \mathbf{z}$ is of length at most $\ell + 1$.

Lemma 32. *Given two vectors* $\mathbf{y}$ *and* $\mathbf{z}$ *of length at most* ℓ, $\mathbf{w} = \mathbf{y} \oplus \mathbf{z}$ *is of length at most* $\ell + 1$

Proof. Let d_i be the carry to the i^{th} position in $\mathbf{y} \oplus \mathbf{z}$. We branch on the value of $d_{\ell+1}$. If $d_{\ell+1} = 0$, then all the bits at positions greater than ℓ in $\mathbf{w}$ are equal to $s_1 \oplus s_2$ and thus the length of $\mathbf{w}$ is at most ℓ. If $d_{\ell+1} = 1$, then if $s_1 \vee s_2 = 0$, $\mathbf{w}(\ell+1) = 1$ and $\mathbf{w}(j) = 0$ for $j > \ell + 1$. Thus the length of $\mathbf{w}$ is at most $\ell + 1$. If $s_1 \vee s_2 = 1$, then it is easy to see that $d_j = 1$ and thus $\mathbf{w}(j) = s_1 \oplus s_2 \oplus 1$ for $j \geq \ell + 1$ and thus the length of $\mathbf{w}$ is at most ℓ. □

Lemma 33. *Let* $\mathbf{y}_1 \cdots \mathbf{y}_k$ *be vectors of length* ℓ *such that* $\lceil \log k \rceil + \ell < m - 1$. *Then* $\mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_k)$ *is of length at most* $\lceil \log k \rceil + \ell$.

Proof. Assume that the statement is true for up to $k/2$ vectors. Then by Corollary 9,

$$\vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_k) - \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{k/2}) \oplus \mathcal{S}(\mathbf{y}_{k/2+1} \cdots \mathbf{y}_k)$$

Now by the induction hypothesis, $\mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{k/2})$ and $\mathcal{S}(\mathbf{y}_{k/2+1} \cdots \mathbf{y}_k)$ are of length at most $\lceil \log k \rceil - 1 + \ell$. Using the previous lemma, we are done. □

Using the observation that for a constant a_1 with bit complexity ℓ, $\mathbf{a}_1 X_1$ is a vector of length ℓ, we have the following corollary.

Corollary 10. *Let* $P = a_1 X_1 + \cdots + a_t X_t$ *be a polynomial with coefficients of bit length at most* ℓ. *Then* $\mathcal{R}(P)$ *is a vector of length at most* $\ell + \lceil \log t \rceil$

Lemma 34. *For any two vectors* $\mathbf{a}$ *and* $\mathbf{b}$ *of length at most* $\ell < m-1$

$$\mathbf{a}(m), \mathbf{b}(m) \vdash (\mathbf{a} \oplus \mathbf{b})(m)$$

Proof. Since $\mathbf{a}$ and $\mathbf{b}$ are of length at most ℓ we have for $m \geq j > \ell$

$$\mathbf{a}(m) \vdash \mathbf{a}(j)$$

$$\mathbf{b}(m) \vdash \mathbf{b}(j)$$

Thus there is no carry beyond position $\ell + 1 < m$ in $\mathbf{a} \oplus \mathbf{b}$ due to our assumptions and thus using identity (2.45) of Lemma 23, it is easy to derive

$$\mathbf{a}(m), \mathbf{b}(m) \vdash (\mathbf{a} \oplus \mathbf{b})(m)$$

□

Since by Lemma 33, the vectors $\mathcal{R}(P_1)$ and $\mathcal{R}(P_2)$ are of length at most $\ell = \lceil \log m_0 \rceil + \lceil \log m_1 \rceil < m-1$ and by Lemma 28, $\vdash \mathcal{R}(P_1 + P_2) - \mathcal{R}(P_1) \oplus \mathcal{R}(P_2)$, we have the following corollary.

Corollary 11. *For any two polynomials* P_1 *and* P_2 *with at most* m_0 *monomials and coefficients of magnitude at most* m_1*,*

$$\mathcal{R}(P_1)(m), \mathcal{R}(P_2)(m) \vdash \mathcal{R}(P_1 + P_2)(m)$$

The following corollary now follows easily from Lemma 26 and the previous lemma.

Corollary 12. *Let* $\mathbf{y}_1 \cdots \mathbf{y}_k$ *be non-negative vectors of length* ℓ *such that* $\lceil \log k \rceil + \ell < m-1$*. Then*

$$\mathbf{y}_1(m), \cdots, \mathbf{y}_k(m) \vdash \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_k)(m)$$

Lemma 35. *Let* $\mathbf{y}$ *and* $\mathbf{z}$ *be two non-negative vectors of length* ℓ *such that* $3\ell < m-1$. *Then*

$$\mathbf{y}(m), \mathbf{z}(m) \vdash \mathcal{SS}(\mathbf{y}, \mathbf{z})(m)$$

Proof. Since $\mathbf{z}$ is non-negative of length ℓ, for $\ell + 1 \leq i \leq m$

$$\mathbf{z}(m) \vdash \mathbf{z}(i)$$

Therefore,

$$\mathcal{SS}(\mathbf{y}, \mathbf{z}) = \mathcal{S}(\mathbf{z}(0)\mathbf{y} \cdots \mathbf{z}(m-1)2^{m-1}\mathbf{y}) = \mathcal{S}(\mathbf{z}(0)\mathbf{y} \cdots \mathbf{z}(\ell)2^{\ell}\mathbf{y})$$

Since each of the vectors $\mathbf{z}(0)\mathbf{y}, \cdots \mathbf{z}(\ell)2^{\ell}\mathbf{y}$ is of length at most 2ℓ and there are ℓ of them, by the previous corollary, we are done.

□

2.D.3 Properties of multiplication

Here we show that multiplication is distributive and can be treated as repeated addition.

Lemma 36. *Distributivity of* $\mathcal{R}$

Let P, P_1, P_2, Q *be polynomials such that* $P = P_1 + P_2$. *Then*

$$\mathcal{R}(PQ) = \mathcal{R}(P_1 Q) \oplus \mathcal{R}(P_2 Q)$$

Proof. Easily follows from Corollary 9 □

The following lemmas show that multiplication is repeated addition.

Lemma 37. *Let y, z be two bits and let* $\mathbf{w}$ *be a vector. Then,*

$$\vdash y\mathbf{w} \oplus z\mathbf{w} - (y \oplus z)\mathbf{w} \oplus H(y,z)2\mathbf{w}$$

Proof. Let $\mathbf{w}_1 = (y \oplus z)\mathbf{w}$ and $\mathbf{w}_2 = H(y,z)2\mathbf{w}$. Let e_i be the carry bit to the i^{th} position in $\mathbf{w}_1 \oplus \mathbf{w}_2$ and let c_i be the carry bit to the i^{th} position in $y\mathbf{w} \oplus z\mathbf{w}$. We will derive by induction that for every i,

$$\vdash (y\mathbf{w} \oplus z\mathbf{w})(i) - (\mathbf{w}_1 \oplus \mathbf{w}_2)(i)$$

$$\vdash e_{i+1} - H(c_i, y\mathbf{w}(i) \oplus z\mathbf{w}(i))$$

This is easy to derive for the case of $i = 1$ since $\mathbf{w}_2(1) = 0$ and thus the first bit on both sides is equal to $(y \oplus z)\mathbf{w}(1)$. Also by (2.45) of Lemma 23, $e_2 = 0$ is derived since $\mathbf{w}_2(1) = 0$ and therefore there is no carry to the second position. Since $c_1 = 0$, $H(c_1, y\mathbf{w}(1) \oplus z\mathbf{w}(1)) = e_2 = 0$. Now assume that we have derived it up to $i-1$ for some $i > 1$. Then we have

$$\vdash e_i - H(c_{i-1}, y\mathbf{w}(i-1) \oplus z\mathbf{w}(i-1))$$

and from the definition of $\mathbf{w_2}$ it is easy to derive

$$\vdash \mathbf{w}_2(i) - H(y\mathbf{w}(i-1), z\mathbf{w}(i-1))$$

Therefore by using Identities (2.42) and (2.45)

$$\vdash e_1 \oplus \mathbf{w}_2(i) - H(c_{i-1}, y\mathbf{w}(i-1), z\mathbf{w}(i-1)) \tag{2.50}$$

$$\vdash e_1 \oplus \mathbf{w}_2(i) - c_i$$

And by Identity (2.44)

$$\vdash H(e_1, \mathbf{w}_2(i)) \tag{2.51}$$

From the above derivations, we now have

$$\vdash e_i \oplus \mathbf{w}_1(i) \oplus \mathbf{w}_2(i) - y\mathbf{w}(i) \oplus z\mathbf{w}(i) \oplus c_i$$

which derives that the i^{th} bits on both sides are equal.

Also, we have by (2.45) of Lemma 23

$$\vdash e_{i+1} - H(e_i, \mathbf{w}_1(i), \mathbf{w}_2(i))$$

By identity (2.42) we have

$$\vdash e_{i+1} - H(e_i, \mathbf{w}_2(i)) \oplus H(\mathbf{w}_1(i), e_i \oplus \mathbf{w}_2(i))$$

and by (2.50) and (2.51)

$$\vdash e_{i+1} - H(y\mathbf{w}(i) \oplus z\mathbf{w}(i), c_i)$$

which continues the induction.

□

Lemma 38. *Let* $\mathbf{y} = [y_{k-1} \cdots y_0]$ *and* $\mathbf{z} = [z_{k-1} \cdots z_0]$ *be two bit vectors of dimension k, let* $\mathbf{w} = \mathbf{y} \oplus \mathbf{z}$ *and let* d_1 *be a constant and* X_1 *be a monomial. Then,*

$$\vdash \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{w}) - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z})$$

Proof. For the base case where **y** and **z** are of dimension one, the above derivation follows easily from the previous lemma. Assume that the statement is derived when **y** and **z** are vectors of

dimension $k-1$. Let $\mathbf{y}_{k-1}$, $\mathbf{z}_{k-1}$, $\mathbf{w}_{k-1}$ denote the corresponding vectors truncated to dimension $k-1$ by dropping the element(s) with the highest index. Let e_i be the carry to the i^{th} position in $\mathbf{y} \oplus \mathbf{z}$, i.e. $\mathbf{w}(i) = y_{i-1} \oplus z_{i-1} \oplus e_i$.

By the definition of $\mathcal{SS}(.)$ and Lemma 26, we derive

$$\begin{aligned} &\vdash \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}) \\ &\quad - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}_{k-1}) \oplus y_{k-1} 2^{k-1} \mathbf{d}_1 X_1 \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}_{k-1}) \oplus z_{k-1} 2^{k-1} \mathbf{d}_1 X_1 \end{aligned}$$

By using associativity (Lemma 24), we have

$$\begin{aligned} &\vdash \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}) \\ &\quad - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}_{k-1}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}_{k-1}) \oplus y_{k-1} 2^{k-1} \mathbf{d}_1 X_1 \oplus z_{k-1} 2^{k-1} \mathbf{d}_1 X_1 \end{aligned}$$

Now using the previous lemma and the induction hypothesis we derive

$$\begin{aligned} &\vdash \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}) \\ &\quad - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}_{k-1} \oplus \mathbf{z}_{k-1}) \oplus (y_{k-1} \oplus z_{k-1}) 2^{k-1} \mathbf{d}_1 X_1 \oplus H(y_{k-1}, z_{k-1}) 2^k \mathbf{d}_1 X_1 \end{aligned}$$

By the definition of $\mathbf{w}_{k-1}$, it is easy to derive

$$\vdash \mathbf{y}_{k-1} \oplus \mathbf{z}_{k-1} - \mathbf{w}_{k-1} \oplus e_k 2^{k-1}\mathbf{1}$$

Now by Lemma 26 and the definition of $\mathcal{SS}(.)$ we have

$$\begin{aligned}\vdash\ & \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}) \\ & - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{w}_{k-1}) \oplus e_k 2^{k-1}\mathbf{d}_1 X_1 \oplus (y_{k-1} \oplus z_{k-1})2^{k-1}\mathbf{d}_1 X_1 \oplus H(y_{k-1}, z_{k-1})2^k \mathbf{d}_1 X_1\end{aligned}$$

By the previous lemma, we can derive

$$\begin{aligned}\vdash e_k 2^{k-1}\mathbf{d}_1 X_1 \oplus (y_{k-1} \oplus z_{k-1})2^{k-1}\mathbf{d}_1 X_1 - \\ (y_{k-1} \oplus z_{k-1} \oplus e_k)2^{k-1}\mathbf{d}_1 X_1 \oplus H(y_{k-1} \oplus z_{k-1}, e_k)2^k \mathbf{d}_1 X_1\end{aligned}$$

Combining this with the above derivation, we have

$$\begin{aligned}\vdash\ & \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{z}) \\ & - \mathcal{SS}(\mathbf{d}_1 X_1, \mathbf{w}_{k-1}) \oplus (y_{k-1} \oplus z_{k-1} \oplus e_k)2^{k-1}\mathbf{d}_1 X_1 \\ & \oplus H(y_{k-1} \oplus z_{k-1}, e_k)2^k \mathbf{d}_1 X_1 \oplus H(y_{k-1}, z_{k-1})2^k \mathbf{d}_1 X_1\end{aligned}$$

Now from identities (2.42) and (2.44) of Lemma 23

$$\vdash \mathcal{SS}(\mathbf{d}_1X_1,\mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1X_1,\mathbf{z})$$
$$-\mathcal{SS}(\mathbf{d}_1X_1,\mathbf{w}_{k-1}) \oplus (y_{k-1} \oplus z_{k-1} \oplus e_k)2^{k-1}\mathbf{d}_1X_1 \oplus H(y_{k-1},z_{k-1},e_k)2^k\mathbf{d}_1X_1$$

Noting that $(y_{k-1} \oplus z_{k-1} \oplus e_k)$ and $H(y_{k-1},z_{k-1},e_k)$ are equal to $\mathbf{w}(k)$ and $\mathbf{w}(k+1)$ respectively, and using the definition of $\mathcal{SS}(.)$ and Lemma 26 we have

$$\vdash \mathcal{SS}(\mathbf{d}_1X_1,\mathbf{y}) \oplus \mathcal{SS}(\mathbf{d}_1X_1,\mathbf{z}) - \mathcal{SS}(\mathbf{d}_1X_1,\mathbf{w})$$

□

Lemma 39. *Let $Q = a'_1X_1 + \cdots + a'_kX_k$ be represented by a bit vector $\mathbf{z} = [z_{m-1}\cdots z_0]$ and let a_0X_0 be a monomial such that the bit length of $a_0a'_i$ is at most $m-1$. Then*

$$\vdash \mathcal{R}(a_0X_0Q) - \mathcal{SS}(a_0X_0,\mathbf{z})$$

Proof. Let $Q_j = a'_1X_1 + \cdots + a'_jX_j$ for $j < k$ and let $\mathbf{z}_j = [z^j_{m-1}\cdots z^j_0]$ be the equal to $\mathcal{R}(Q_j)$. Assume that we have proved the above statement for Q_j, $j < k$. Then by Lemma 26, $\vdash \mathbf{z} - \mathbf{z}_{k-1} \oplus \mathbf{a}'_kX_k$. Therefore by Lemma 38 we have

$$\vdash \mathcal{SS}(\mathbf{a}_0X_0,\mathbf{z}) - \mathcal{SS}(\mathbf{a}_0X_0,\mathbf{z}_{k-1}) \oplus \mathcal{SS}(\mathbf{a}_0X_0,\mathbf{a}'_kX_k)$$

Since the bit length of $a_0a'_i$ is at most $m-1$, $\mathcal{SS}(\mathbf{a}_0X_0,\mathbf{a}'_kX_k) = \mathcal{R}(a_0a'_kX_0X_k)$ by definition and by induction,

$$\vdash \mathcal{SS}(\mathbf{a}_0X_0,\mathbf{z}_{k-1}) - \mathcal{R}(a_0X_0Q_{k-1})$$

Therefore we have

$$\vdash \mathcal{SS}(\mathbf{a}_0 X_0, \mathbf{z}) - \mathcal{R}(a_0 X_0 Q_{k-1}) \oplus \mathcal{R}(a_0 a'_k X_0 X_k)$$

which is equal to $\mathcal{R}(a_0 X_0 Q_k)$ by the Distributivity of $\mathcal{R}$.

□

Lemma 40. *Let P and Q be two polynomials, represented by bit vectors $\mathbf{y}_0$ and $\mathbf{z} = [z_{m-1} \cdots z_0]$, with at most m_0 monomials and coefficients bounded by m_1 in absolute value. Then,*

$$\vdash \mathcal{R}(PQ) - \mathcal{SS}(\mathbf{y}_0, \mathbf{z})$$

Proof. Let $P = a_1 M_1 + \cdots + a_k M_k$, $Q = a'_1 M_1 + \cdots + a'_k M_k$ and let P_j be the sum of the first $j < k$ terms of P . Let $\mathbf{y}_i$ denote the bit vector $2^i \mathcal{R}(P)$. Then $\mathcal{SS}(\mathbf{y}, \mathbf{z}) = \mathcal{S}(z_0 \mathbf{y}_0 \cdots z_{m-1} \mathbf{y}_{m-1})$.

It is easy to derive for vectors $\mathbf{a}$ and $\mathbf{b}$ and any i

$$\vdash 2^i(\mathbf{a} \oplus \mathbf{b}) - 2^i \mathbf{a} \oplus 2^i \mathbf{b}$$

Now by a simple induction using Lemma 26 we derive

$$\vdash 2^i \mathcal{S}(\mathbf{a}_1 M_1 \cdots \mathbf{a}_k M_k) - \mathcal{S}(2^i \mathbf{a}_1 M_1 \cdots 2^i \mathbf{a}_k M_k)$$

$$\vdash \mathbf{y}_i - \mathcal{S}(2^i \mathbf{a}_1 M_1 \cdots 2^i \mathbf{a}_k M_k)$$

Let $\mathbf{y}_i^j = \mathcal{S}(2^i \mathbf{a}_1 M_1 \cdots 2^i \mathbf{a}_j M_j)$ for $j < k$. By Lemma 26, $\mathbf{y}_i = \mathbf{y}_i^{k-1} \oplus 2^i \mathbf{a}_k M_k$. Therefore we have

$$\vdash \mathcal{S}(z_0 \mathbf{y}_0 \cdots z_{m-1} \mathbf{y}_{m-1}) - \mathcal{S}(z_0 \mathbf{y}_0^{k-1} \oplus z_0 \mathbf{a}_k M_k \cdots z_{m-1} \mathbf{y}_{m-1}^{k-1} \oplus z_{m-1} 2^{m-1} \mathbf{a}_k M_k)$$

By repeated applications of Lemma 26 we can derive

$$\vdash \mathcal{S}(z_0\mathbf{y}_0\cdots z_{m-1}\mathbf{y}_{m-1}) - \mathcal{S}(z_0\mathbf{y}_0^{k-1}\cdots z_{m-1}\mathbf{y}_{m-1}^{k-1}) \oplus \mathcal{S}(z_0\mathbf{a}_kM_k\cdots z_{m-1}2^{m-1}\mathbf{a}_kM_k)$$

By the definition of $\mathcal{SS}(.)$ we have $\mathcal{SS}(\mathbf{a}_kM_k,\mathbf{z}) = \mathcal{S}(z_0\mathbf{a}_kM_k\cdots z_{m-1}2^{m-1}\mathbf{a}_kM_k)$ and by Lemma 39 we have

$$\vdash \mathcal{SS}(\mathbf{a}_kM_k,\mathbf{z}) - \mathcal{R}(\mathbf{a}_kM_kQ)$$

and by induction on k we have

$$\vdash \mathcal{SS}(\mathbf{y}_0^{k-1},\mathbf{z}) - \mathcal{R}(P_{k-1}Q)$$

Thus we derive

$$\vdash \mathcal{S}(z_0\mathbf{y}_0\cdots z_{m-1}\mathbf{y}_{m-1}) - \mathcal{R}(P_{k-1}Q) \oplus \mathcal{R}(\mathbf{a}_kM_kQ)$$

The lemma now follows from Distributivity of $\mathcal{R}$.

Chapter 2 contains material from "The surprising power of constant depth Algebraic Proofs." by Russell Impagliazzo, Sasank Mouli, and Toniann Pitassi.which is published at the Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. 2020.

□

Chapter 3

Lower bounds for Polynomial Calculus with extension variables over finite fields

3.1 Introduction

Propositional proof complexity, started by the seminal work of Cook-Reckhow [CR79] is a field of study analogous to Boolean circuit complexity, and asks the following question: given a formal proof system dealing with propositional formulae, what Boolean tautologies are hard to prove in this system? The ultimate goal of this program is to come up with tautologies that are hard for any proof system that has a polynomial time verifier, hence showing that $\mathsf{coNP} \not\subseteq \mathsf{NP}$. So far progress has been made only for proof systems which are restricted in their capacity to reason, such as Resolution [Hak85],[BSW99] and Bounded-depth Frege [Ajt94], [**?**],[**?**].

The next frontier is to obtain lower bounds for the system $\mathsf{AC}^0[p]$-Frege. Since for the analogous circuit model, Razborov [**?**] and Smolensky [**?**] obtained lower bounds through connections to algebraic objects, this suggests the study of *Algebraic Proof Systems*. Beame et.al. [**?**] introduced and showed lower bounds for the algebraic proof system *Nullstellensatz* where lines are low degree polynomials over a field. Clegg et. al [CEI96] introduced the algebraic

system *Polynomial Calculus(*PC*)*, which is a dynamic generalization of Nullstellensatz. Lower bounds for Polynomial Calculus were obtained in many works (e.g., [Raz98a, BGIP01, AR01, GL10, MN15]).

After what seemed to be reasonable progress with Algebraic Proof Systems, progress towards the frontier of $\mathsf{AC}^0[p]$-Frege is still stalled for various reasons. For one, the proof systems described above are not strong enough to simulate $AC^0[p]$-Frege. Also, the lower bound techniques used for the above lower bounds are based on random restrictions, and it is well known that modular counting gates are immune to such techniques. Therefore, we need to aim towards lower bounds for systems strong enough to simulate $AC^0[p]$-Frege. We also need lower bound techniques that are not just random-restriction based. On both fronts, there has been recent progress. Grigoriev and Hirsch [GH03] introduced the proof system constant-depth PC and showed that it simulates $\mathsf{AC}^0[p]$-Frege at a proportional depth. Raz and Tzameret [**?**] showed that constant-depth PC at depth 3 over the field of Rational numbers surprisingly simulates the semi-algebraic proof system Cutting Planes with bounded coefficients. [IMP20] obtained many more surprising results, and showed that Depth-43-PC (over a large enough extension of a finite field $\mathbb{F}_p$) simulates $AC^0[q]$-Frege for a different prime q, Cutting Planes with unbounded coefficients, and also the Sum-of-Squares proof system. They also showed that Depth-d-PC simulates TC^0-Frege at a proportional depth. It therefore makes sense to aim for lower bounds for the stronger system Depth-d-PC, for increasingly large constants d. In this direction, Sokolov [Sok20] obtained lower bounds for the system PC over $\mathbb{F}_3$ where extension variables of the form $z_i = 2x_i - 1$ are allowed to be introduced (hence making them take values in the set $\{+1,-1\}$).

In this work, we generalize the methods of Sokolov to show lower bounds for PC with up to $N^{2-\varepsilon}$ extension variables which can depend on up to $\kappa = O(\log N)$ original variables (where N is the number of variables in the tautology). [Ale21] obtained stronger lower bounds for Polynomial Calculus with extension variables over the reals, but since we work over finite fields our results are incomparable. Also, their tautology is a variant of subset sum with large

coefficients, which cannot be defined well over finite fields.

Theorem 16 (high-end)**.** *There is a family of CNF tautologies $\psi_{N,\kappa,M}$ on N variables with $poly(N)$ clauses of width $O(logN)$ so that for any $M = Npolylog(N)$ and $\kappa \geq 1$, and prime p, there is a function $S(N) \in 2^{\Omega(N/polylog(N))}$ so that any* PC *refutation of $\psi_{N,\kappa,M}$ together with any M κ-local extensions over $\mathbb{F}_p$ requires size $S(N)$.*

Theorem 17 (low-end)**.** *For the same family of tautologies above, for any prime p, there are $0 < \alpha, \beta, \gamma < 1$, with $\gamma < 1 - \alpha - \beta$ so that, for $M = N^{1+\alpha}, \kappa = \beta logN$, and $S = exp(N^{\gamma})$, any* PC *refutation of $\psi_{N,\kappa,M}$ together with any M κ-local extensions over $\mathbb{F}_p$ requires size $S(N)$.*

3.1.1 Related Work

Our primary goal in this work is to prove lower bounds for low-depth IPS refutations for unsatisfiable CNF formulas, over finite fields, since this is the setting where algebraic lower bounds are closely connected to proving lower bounds for $AC^0[p]$-Frege systems, a major and longstanding open problem in proof complexity.

The work that inspired us and that is most related to our result is the recent paper by Sokolov [Sok20], proving exponential lower bounds on the size of PC refutations of CNF formulas over $\mathbb{F}_3$, where the variables take on values in $\{1, -1\}$. We generalize Sokolov's result to hold over any finite field, even with the addition of superlinear many extension variables, each depending arbitrarily on a small number of original variables. Thus our result can be alternatively viewed as making progress towards proving exponential lower bounds for depth-3 IPS, for a family of CNF formulas.

We note that for more general unsatisfiable polynomial equations that are *not* translations of CNF formulas, several recent papers prove much stronger results over the rationals (but are incomparable to our main result). First, [**?**] proved lower bounds for subsystems of IPS by restricted classes of circuits, including low-depth formulas, multilinear formulas and read-once

oblivious branching programs. Secondly, Alekseev [Ale21] proved exponential lower bound on the bit complexity of constant-depth IPS proofs over the rationals, and a recent paper by Andrews and Forbes [**?**] proves quasipolynomial lower bounds on the circuit size of constant-depth IPS proofs for a different family of formulas, over the rationals. However, for a variety of reasons, all of these lower bounds do not imply nontrivial lower bounds in the propositional setting and thus they are incomparable to our main result. First, these results do not hold in the setting of finite fields. Secondly, the particular choice of hard polynomials are inherently nonboolean: [**?**, Ale21] use the subset sum principle which when translated to a propositional statement is no longer hard, and the hard polynomials in [**?**] have logarithmic depth.

3.1.2 Our Result: Proof Overview

The standard way of proving size lower bounds for PC for an unsatisfiable formula F for Boolean-valued variables dates back to the celebrated superpolynomial lower bounds for Resolution [Hak85, BSW99], where the basic tool is to reduce size lower bounds to degree lower bounds (or in the case of Resolution, size to clause-width) by way of either a general size-depth tradeoff, or by a more general random restriction argument. At a high level, both methods iteratively select a variable that occurs in a lot of high-degree terms, set this variable to zero (to kill off all high-degree terms containing it), while also ensuring (possibly by setting additional variables) that F remains hard to refute after applying the partial restriction. After applying this size-to-degree reduction, the main technical part is to prove degee lower bounds for the restricted version of F.

Unfortunately over the $\{-1,1\}$ basis, the size to degree reduction breaks down. In fact, no generic reduction to degree can exist since random XOR instances over this basis require linear degree but have polynomial size PC refutations over GF_2. Moreover, we lacked *any* method for proving PC lower bounds for unsatisfiable CNFs over the basis $\{-1,1\}$. and more generally over an arbitrary linear transformation of the variables. In [IMP20], we highlighted this as an open

problem, noting that it is a necessary step toward proving superpolynomial $AC^0[2]$-Frege lower bounds, a major open problem in proof complexity.

Recently, Sokolov [Sok20] made significant progress by proving exponential lower bounds for PC (as well as for SOS) for random CNF formulas over the domain $\{-1,1\}$, by developing new formula-specific techniques to reduce size to degree over this domain. As this is the starting point for our work, we begin by describing the main method in [Sok20] for reducing size to degree for certain families of formulas over $\{-1,1\}$.

Let Π be an alleged PC refutation of F of small size which includes the axioms $w^2 = 1$ for all variables w. The first step in Sokolov's argument is to show how to remove all high degree terms containing a particular variable w, provided that w is *irrelevant* – meaning that it does not occur in any of the initial polynomials other than the equation $w^2 = 1$. Intuitively, we want to show that if our unsatisfiable system of polynomial equations doesn't contain w, then we should be able to eliminate w altogether from the refutation. To show this, Sokolov writes each line q in the refutation as $q_0 + q_1 w$, and proves by induction that if we replace each line q by the pair of lines q_0, q_1, then it is still a valid refutation of F (and no longer contains w). While the split operation removes w from the proof, it doesn't kill off high degree terms. The crucial insight is that although this doesn't directly kill off high degree terms, a slightly different measure of degree (called quadratic degree) can be used instead, since removing w via the split operation removes all high quadratic degree terms that w contributed to, and secondly low quadratic degree implies low ordinary degree. The second and easier step in Sokolov's argument uses specific expansion properties of F to shows that for any variable w, there exists a small restriction ρ (to some of the other variables) such that w becomes irrelevant under ρ.

Our main theorem significantly generalizes Sokolov's lower bound by proving exponential lower bounds for an unsatisfiable CNF formulas F, even when we allow the axioms P to contain superlinear many extension axioms, provided that each has small support. Note that the variables of F are Boolean, but the extension variables are not restricted to being Boolean. In particular,

they may be *nonsingular*, meaning that setting the variable to zero falsifies one of the initial polynomials. Intuitively, a nonsingular variable w cannot be set to 0, so we will handle them in a similar manner to Sokolov, by first isolating w, and then generalizing the split operation in order to kill off all large quadratic degree terms that contain w. However, dealing with a general set of extension axioms presents new technical challenges that we address next.

Our first idea is to design the unsatisfiable formula F carefully so that we can force variables to be irrelevant in a more modular way. Specifically, let $F'(x_1,\ldots,x_n)$ be an expanding unsatisfiable k-CNF formula with $m = O(n)$ clauses, such that any subset of $m' = \delta m$ clauses is unsatisfiable and requires proofs of large PC degree. We define an unsatisfiable formula F (based on F') that intuitively states that there is a subset S of $m' = \delta m$ clauses of F' (as chosen by new selector variables y) that is satisfiable. We will prove lower bounds on the set of constraints $F \cup E$, where E is an arbitrary set of extension axioms satisfying the conditions mentioned earlier. In order to make a variable of F' irrelevant, we will simply make sure that our eventual assignment to the selector variables (y) avoids constraints of F' that contain this variable.

A second challenge that we face (that doesn't come up in Sokolov's proof) is that extension variables start off as either singular or nonsingular (recall that singular variables can take on the value zero without falsifying any of the initial or extension axioms, while nonsingular variables cannot), but can change status after applying a restriction. For example, suppose E includes the extension axiom $z = x_1x_2 + x_1$. Then z is singular (since we can set $x_1 = x_2 = 0$), but if we set $x_1 = 1$, then z becomes nonsingular. In order to deal with this dynamically changing status of variables, our notion of quadratic degree must pay attention to which category each of the extension variables is in at any particular time, and make sure that we do not lose progress that was made earlier due to variables changing from singular to nonsingular. Fortunately we observe that variables can only change unidirectionally, from singular to nonsingular, and this is crucial for arguing that we can iteratively kill off large quadratic degree terms with respect to both types of variables in a particular order so that we continually make progress.

Finally, we also have to generalize Sokolov's split operation, which was previously defined only for $\{-1,1\}$ variables. We give a generalization of how to do the split for arbitrary valued variables.

3.2 Preliminaries

Definition 26 (Polynomial Calculus)**.** *Let $\Gamma = \{P_1 \cdots P_m\}$ be a set of polynomials in variables $\{x_1 \cdots x_n\}$ over a field $\mathbb{F}$ such that the system of equations $P_1 = 0 \cdots P_m = 0$ has no solution. A Polynomial Calculus refutation of Γ is a sequence of polynomials $R_1 \cdots R_s$ where $R_s = 1$ and for every ℓ in $\{1 \cdots s\}$, $R_\ell \in \Gamma$ or is obtained through one of the following derivation rules for $j,k < \ell$*

$R_\ell = \alpha R_j + \beta R_k$ *for* $\alpha, \beta \in \mathbb{F}$

$R_\ell = x_i R_k$ *for some* $i \in \{1 \cdots n\}$

The size of the refutation is $\sum_{\ell=1}^{s} |R_\ell|$, where $|R_\ell|$ is the number of monomials in the polynomial R_ℓ. The degree of the refutation is $\max_\ell deg(R_\ell)$.

Definition 27 (PC with extension variables)**.** *Let $\Gamma = \{P_1 \cdots P_m\}$ be a set of polynomials in variables $\{x_1 \cdots x_n\}$ over a field $\mathbb{F}$ such that the system of equations $P_1 = 0 \cdots P_m = 0$ has no solution. Let $z_1 \cdots z_k$ be new variables defined over $\{x_1 \cdots x_n\}$ by $z_j = Q_j(x_1 \cdots x_n)$. A* PC *refutation of Γ with extension variables $z_1 \cdots z_k$ is a Polynomial Calculus refutation of the set $\Gamma' = \{P_1 \cdots P_m, z_1 - Q_1, \cdots, z_k - Q_k\}$ of polynomials over $\{x_1 \cdots x_n\}$ and $\{z_1 \cdots z_k\}$.*

The size of such a refutation is the size of the Polynomial Calculus refutation of Γ'

Definition 28 (Refutation of a k-CNF in Polynomial Calculus)**.** *Since we are working with Polynomial Calculus, a tautology in clausal form has to be translated into a set of polynomials over a field. We work over $\mathbb{F}_p$ for prime $p > 2$ and use the standard* PCR *translation of CNFs into polynomials: for each variable x occuring in the CNF, we have two associated variables x and $\bar{x}$ (representing x and its negation respectively). Each clause C is converted into an associated*

monomial in the natural way. For example, if $C = (x_1 \vee \neg x_2 \vee x_3)$*, then the corresponding polynomial is* $\bar{x}_1 x_2 \bar{x}_3 = 0$*. In addition, for every variable* $w, \overline{w}$*, we include the boolean axioms* $w^2 - w = 0$*,* $\overline{w}^2 - \overline{w} = 0$*, as well as the axiom* $w\overline{w} = 0$*. .*

3.3 The Hard Formulas

We distinguish between the case $p = 2$ and the case $p > 2$, and concentrate on the latter. This is because the case $p = 2$ does not require any new technical ideas, and we can pick from a large number of known hard tautologies for this case, such as random *CNF*'s. Over $\mathbb{F}_2$, every extension variable is zero-one valued, and so standard size-degree tradeoffs pertain even with respect to extension variables. Also, k-local extension variables can change the degree by at most a factor of k. Since to use the size-degree tradeoffs, the degree must be at least the square root of the number of variables, this immediately gives a lower bound tolerating close to a quadratic number of local extension variables for any tautology requiring linear degree, giving us our claimed results.

Note, however, that over any field with $p > 2$, the Tseitin tautologies require linear degree but have polynomial sized proofs with a linear number of extension variables, so high degree is not sufficient when $p > 2$. So in this case, we need a new type of hard tautology. Below, we describe these tautologies.

We start with any unsatisfiable CNF formula such that

a) Any small set of variables appear in a small fraction of the axioms

b) Any large enough subset of axioms is unsatisfiable, and requires linear PC degree to refute.

For concreteness, we fix the following unsatisfiable CNF obtained by generating sufficiently many random parities.

First we'll show that a random regular bipartite graph has good boundary expansion. This has been used implicitly in other works ([CS88], [BKPS02]), but we could not find a clean statement to cite, so for completeness we state and prove it here. Let $G = (L, R, E)$ be a bipartite graph, and let $A \subseteq R$. The *boundary* for A, $\partial(A)$, is the set of vertices x in L so that $|N(x) \cap A| = 1$, i.e., vertices with a unique neighbor in A. A bipartite graph is (d,k) regular if every vertex in L has degree d and every vertex in R has degree k. In this case, for $n = |L|, m = |R|$, we have $dn = km$.

Theorem 18. *Let d, k, n, m be positive integers with $dn = km$, $k \geq 12$. Then with high probability for a random (d,k) regular bipartite graph with $|L| = n, |R| = m$, for all $A \subset R$, $|A| < n/(e^6 k^2)$, we have $\partial(A) \geq k|A|/2$.*

Proof. Let $N(A)$ be all the neighbors of A. Since the total degrees of vertices in A is $k|A|$, and each element of $N(A) - \partial(A)$ is contingent on two such edges, $k|A| \geq 2(|N(A)| - |\partial(A)|) + |\partial(A)$, or $\partial(A) \geq 2|N(A)| - k|A|$. We will show that with high probability for all such A, $|N(A)| > 3k|A|/4$, and hence $\partial(A) \geq k|A|/2$.

If not, there are sets $A \subset R$ and $B \subset L$ so that $N(A) \subseteq B$ and $|B| = 3k|A|/4$. We will bound the probability that this is true for fixed sets A, B and then take a union bound. We can view picking a random (d,k) bipartite graph as picking a random matching between d half-edges adjacent to each $x \in L$ and k such half-edges adjacent to each $y \in R$; if a half edge for x is matched to a half-edge for y, it forms an edge between x and y.

We can form this matching by going through the half edges for nodes in R and for each randomly selecting an unmatched half-edge for some node in L. We start with the edges for A in an arbitrary order. If we condition on all previous neighbors for A being in B, the number of half-edges left still available for B is less than $d|B|$, whereas the number for $\overline{B}$ stays at exactly $d(n - |B|)$. Thus, the conditional probability that the next edge formed is also in B is at most $|B|/n$, and we do this for each of $k|A|$ edges, meaning the probability that all neighbors are in B is at most $(|B|/n)^{k|A|}$.

Now, for a fixed $|A|$ and setting $|B| = 3k|A|/4$, we take the union bound over all subsets A and B. This gives a total probability of failure for some set A of size a as :

$$\binom{m}{a}\binom{n}{3ka/4}(3ka/4n)^{ka}$$

$$\leq (em/a)^a(4en/3ka)^{3ka/4}(3ka/4n)^{ka}$$

$$\leq (em/a)^a(e^3ka/n)^{ka/4} = (ekn/da)^a(e^3ka/n)^{ka/4} = (e^{3k/4+1}a^{k/4-1}k^{k/4+1}/dn^{k/4-1})^a$$

Since we are assuming $a < n/(e^6k^2)$, the base in the above expression is at most

$$e^{3k/4+1}(n/e^6k^2)^{k/4-1}k^{k/4+1}/dn^{k/4-1}$$

$$= e^{7-3k/4}k^{3-k/4}/d$$

which for $k \geq 12$ is bounded below e^{-2}, meaning the probability of such a bad set existing is exponentially small in a, and the probability of such a bad set existing for any a is less than $1/2$.

□

Definition 29. *For a Boolean vector $X = \{x_1, \ldots, x_n\}$, we define $\mathcal{L}_{n,m,k_1,k}(X)$ to be a distribution over k-CNF formulas over n variables $X = \{x_1, \ldots, x_n\}$ obtained by selecting m parities, where each parity is represented by a node on the right of a bipartite graph $G(L,R)$ with left degree bounded by k_1 and right degree bounded by k chosen uniformly at random from all such graphs.*

Lemma 41. *Let $F_{n,k}$ be a tautology given by $AX = b$ over variables $X = \{x_1, \ldots, x_n\}$ where A is the adjacency matrix of a graph drawn at random from $\mathcal{L}_{n,m,k_1,k}$ where $m = 10n$, for large enough constants $k_1, k > 0$, and b is chosen randomly. Then the following hold with high probability for a small enough $\varepsilon > 0$:*

a) Any subset of a $(1-\varepsilon)$-fraction of the equations in $F_{n,k}$ is unsatisfiable

b) Any subset of a $(1-\varepsilon)$-fraction of the equations in $F_{n,k}$ requires PC *degree $c_2(n)$ to refute, for some $c_2 > 0$.*

Proof. a) The probability that a set of $(1-\varepsilon)10n$ random parities (i.e. for a random choice of b) is satisfiable is at most 2^{-9n} for a small enough ε. The probability that any such subset of $F_{n,k}$ is satisfiable is therefore at most $2^{(-n(9-10H(\varepsilon)))}$, which is exponentially small for a small enough ε, where $H()$ is the binary entropy function.

b) This follows directly from [AR01], Theorem 3.8 and Theorem 4.4, since by Theorem 18 the graph with adjacency matrix A has good expansion with high probability.

□

We now want to compose $F_{n,k}$ with a function we call SELECT, which encodes a complete bipartite graph such that nodes on the left represent equations of $F_{n,k}$ of which a large enough subset is selected by the nodes on the right. In order to eliminate an equation from being selected on the right, we add it to a running list of bad equations, and reduce the proof by the set of assertions stating that no node on the right can be assigned this equation. Conversely, by substituting a complete assignment for the variables of SELECT, we would like to be able to select sufficiently many equations from any large enough subset on the left.

We now define our tautology below.

Definition 30. *Let $F_{n,k}(X) = \{E_i \mid i \in [m]\}$ and ε be as in Lemma 41, $m = 10n$, $m' = (1-\varepsilon/2)m$ and Let $Y = \{y_{ij}, i \in [m'], j \in [\log m]\}$ and let Y_i be defined appropriately. For bits $b_1 \cdots b_{\log m}$ let $Y_i \neq b_1 \cdots b_{\log m}$ represent the formula $\prod_j (y_{ij} - b_j \oplus 1)$. Then $F_{n,k}^{SEL}$ is the following set of clauses in $O(n)$ variables $X \cup Y$.*

$$Y_i \neq b_1 \cdots b_{\log m} \vee E_{b_1 \cdots b_{\log m}}$$

$$Y_i \neq b_1 \cdots b_{\log m} \vee Y_{i'} \neq b_1 \cdots b_{\log m}$$

for $i \neq i'$

In the above definition, we refer to the variables $Y_i = y_{i1} \cdots y_{i\log m}$ as the i^{th} pigeon. Thus the axioms can be interpreted as the following two statements:

1. If the i^{th} pigeon maps to the string $b_1 \cdots b_{\log m}$ for any i, the equation $E_{b_1 \cdots b_{\log m}}$ is true.

2. For any $b_1 \cdots b_{\log m}$ and indices i, i', either the i^{th} pigeon does not map to it or the i'^{th} pigeon does not map to it.

Since we are working with Polynomial Calculus, the above CNF formula has to be translated into a set of polynomials, as described in Definition 28.

Definition 31. *A locality* κ *extension variable is a new variable* z *together with a single polynomial defining constraint* $z = q(w_{i_1}, ..w_{i_k})$ *for some polynomial* q *and* κ *original variables* $w_{i_1}, \ldots, w_{i_k} \in X \cup Y$.

Definition 32. *Let* $\psi_{N,\kappa,M}(W)$ *denote the unsatisfiable formula* $F^{SEL}_{n,k}$ *with* M *extension axioms* Z *of locality* κ*, and where* $W = X \cup Y \cup Z$*, and* $|W| = N$.

3.4 The Lower Bound

3.4.1 Technical Proof Overview.

We start with a PC refutation Π of $\psi_{N,\kappa,M}(W)$ over $\mathbb{F}_p$. Conventionally, proof size lower bounds are reduced to degree lower bounds, a single step of which involves finding a variable that occurs in a large fraction of high degree terms of the proof and setting it to zero. In our setting, if the latter turns out to be an extension variable, z with extension axiom $z = f(X,Y)$ it may be *nonsingular* meaning that setting $z = 0$ will falsify the extension axiom for z. In this case, we cannot simply eliminate the high degree terms containing z by setting $z = 0$. Sokolov [Sok20] introduced *Quadratic degree* as a measure to be used instead of degree in such cases and showed

that a refutation of low quadratic degree can be turned into one of low degree. Quadratic degree essentially measures the maximal degree of the *square* of each polynomial P occurring in the proof. Sokolov also introduced an operation *Split* that acts on a proof line by line in order to reduce quadratic degree in the special case of nonsingular variables that always take on values in ± 1.

In our case, we have to deal with the case of both singular as well as nonsingular variables, and where the nonsingular variables can depend arbitrarily on a logarithmic number of original variables. To accomplish this, we give a procedure that reduces reduce both ordinary degree (for singular variables occurring in the proof) as well as quadratic degree (for nonsingular variables).

The first phase of our procedure deals with eliminating the set S of all high *singular* degree terms – that is, terms that contain many singular variables. This is handled in a standard way, by finding a singular variable w occurring in many terms of S, and applying the restriction σ which sets $w = 0$. Then in order to ensure that the properties of the tautology remain intact after applying σ, we maintain a list B of "bad" axioms and modify the formula and proof (using **X-cleanup**, **Y-cleanup** and Lemma 50) so that the selector variables cannot map to any axiom in B. In this case, we add to B any axiom that is affected by σ and modify the proof so that the Y-variables avoid mapping to any axiom in B.

The second more difficult phase of our procedure deals with removing all terms of large quadratic degree from the proof. Assume that w is some *nonsingular* variable occuring in many terms of high quadratic degree. Since all variables in $X \cup Y$ are singular, w must be an extension variable z with corresponding extension axiom $z = f(X,Y)$. First, we apply a partial assignment σ to all but one X-variable of $f(X,Y)$ so that $f(X,Y)|_\sigma$ reduces to a linear function of a single X-variable, x (extension variables that only depend on the selector variables Y are inconsequential since we substitute a complete assignment to the variables Y at the end of our procedure). After applying σ (which sets a small number of both X and Y variables), we apply cleanup procedures to get rid of all axioms that were affected by σ, and also those that contain x. As in the first

phase, this involves updating our list B of bad axioms, and as well as modifying the proof, using **X-cleanup**, **Y-cleanup** procedures together with Lemma 50.

Now that the axioms are free of both x and z, our main technical contribution (Lemma 49) significantly generalizes Sokolov's Split operation in order to eliminate z from the proof (thus making z irrelevant). This in turn yields a near-constant factor reduction in the number of large quadratic degree terms. By iterating this process, we obtain a refutation of a reduced version of $\psi_{N,\kappa,M}(X)$ of low quadratic degree. Crucially, our procedure for reducing quadratic degree does not increase the singular degree, and thus at the end of the second phase, we have extracted a proof (of a reduced but still hard formula) that has low singular degree as well as low quadratic degree. Then by Lemma 43, this in turn implies a proof of small overall degree.

Finally, we argue that we can substitute an assignment for the selector variables Y (since each step adds only a small number of axioms to the bad set B, and therefore the total size of B is at most a constant fraction of the original axioms) in order to obtain a low degree refutation of a large subset of $F_{n,k}$, which gives us a contradiction.

3.4.2 Quadratic Degree, and Removing Irrelevant Variables

Recall that p is the characteristic of the underlying field.

Definition 33 (Singular degree)**.** *For an extension variable z, we say that it is singular if it can take the value zero. Else we say that it is non-singular (any non-extension variable w is singular by default since $w^2 = w$ holds). By Lemma 42 below, a nonsingular variable implies $z^{p-1} = 1$. For a term t, let the singular degree $sing(t)$ denote the number of singular variables in t.*

Lemma 42. *For z non-singular, we can derive $z^{p-1} = 1$ from the extension axiom for z.*

Proof. Let q be any multi-linear polynomial in the original variables and let $z = q$ be the defining equation for z. Look at $q^{p-1}(X,Y)$. Since q is never $0 \mod p$ for Boolean inputs, this is always equal to 1 for Boolean inputs. But since every function from Boolean inputs to the field has a

unique representation as a multi-linear function, when we make q^{p-1} multi-linear, it must be the identically 1 polynomial. Then $z^{p-1} = q^{p-1}$ is derivable from the defining axiom, which means $z^{p-1} - 1$ is derivable. □

Definition 34. *For a term t and a variable w (or its negated version $\bar{w}$), $\deg(t,w)$ is equal to the degree of w in t. Note that since we are over a finite field of characteristic p, $\deg(t,w) \leq p$. If w is nonsingular, then $w^{p-1} = 1 \mod p$, so $\deg(t,w) \leq p-1$. For a term t the overall degree of t, $\deg(t)$, equals $\sum_{w \in W} \deg(t,w)$.*

The next definition is a generalization/modification of Sokolov's definition of quadratic degree for the more general scenario where the proof contains both singular and nonsingular variables.

Definition 35 (Quadratic degree)**.** *For a pair of terms t_1, t_2, and a variable w (or its negated version $\bar{w}$), we define the weight of w with respect to t_1 and t_2, $Qdeg(t_1,t_2,w)$ as follows. If $w \in X \cup Y$, then $Qdeg(t_1,t_2,w) = 1$ if w occurs in at least one of t_1 or t_2 and zero otherwise; if w is an extension variable, then $Qdeg(t_1,t_2,w) = 1$ if and only if $\deg(t_1,w) \neq \deg(t_2,w)$ and zero otherwise. The overall weight of the pair t_1,t_2, $Qdeg(t_1,t_2)$, is equal to $\sum_{w \in W} Qdeg(t_1,t_2,w)$. The weight of a polynomial P is equal to the maximum weight over all pairs (t_1,t_2) such that $t_1,t_2 \in P$. The quadratic degree of P is defined as the maximum weight of P. For a proof Π, the quadratic degree is the maximum quadratic degree over all polynomials $P \in \Pi$.*

Definition 36 (Q)**.** *For a polynomial P, $Q(P) = \{(t_1,t_2) | t_1,t_2 \in P\}$. For a pair of polynomials P_1 and P_2, $Q(P_1,P_2) = \{(t_1,t_2) | t_1 \in P_1, t_2 \in P_2$ or vice versa$\}$. $Q(\Pi) = \cup_{P \in \Pi} Q(P)$.*

Definition 37 ($\mathcal{H}_d$)**.** *For a proof Π, let $\mathcal{H}_d(\Pi)$ denote the set of pairs (t_1,t_2) of high quadratic degree. That is, $\mathcal{H}_d(\Pi)$ is the set of pairs of terms (t_1,t_2) such that t_1,t_2 both occur in P for some polynomial $P \in \Pi$, and $Qdeg(t_1,t_2) \geq d$.*

Observation 1. *Substitution does not raise the quadratic degree, i.e. if P is a polynomial, x is a variable occuring in it and $a \in \mathbb{F}_p$ then the quadratic degree of $P_{|_{x=a}}$ is at most that of P.*

Proof. This follows from the fact that for any two terms $t_1, t_2 \in P$, $Qdeg(t_1, t_2, w)$ remains unchanged if w is different from x, and decreases if $w = x$. □

The following is a generalized version of the argument from [Sok20] that shows how to convert a proof with low quadratic degree to one with low degree.

Lemma 43. *If a set of unsatisfiable polynomials $\mathcal{F}$ of degree d_0 has a* PC *refutation of quadratic degree and singular degree at most d, then it has a* PC *refutation of degree $O(p^2 \max(d, d_0))$.*

Proof. We first observe that for any two terms t_1, t_2, $deg(t_1 t_2^{p-2}) \leq p \cdot (Qdeg(t_1, t_2) + sing(t_1) + sing(t_2))$. This is due to the following. Note that any singular variable that appears in either t_1 or t_2 appears in $t_1 t_2^{p-2}$ with degree at most p. Thus, the degree of singular variables in $t_1 t_2^{p-2}$ is at most p times $sing(t_1) + sing(t_2)$. For a nonsingular extension variable z that occurs in t_1 and t_2 such that $deg(t_1, z) = deg(t_2, z)$, $deg(t_1 t_2^{p-2}, z) = Qdeg(t_1, t_2, z) = 0$. Any other nonsingular variable that occurs in at least one of t_1 and t_2 has $deg(t_1 t_2^{p-2}, x) \leq p - 1$ and $Qdeg(t_1, t_2, x) = 1$. Therefore the degree of nonsingular variables in $t_1 t_2^{p-2}$ is at most $p - 1$ times $Qdeg(t_1, t_2)$.

From the above observation, it suffices to prove the following: Let F be a set of unsatisfiable polynomials of degree d_0 with a PC refutation, Π, over $\mathbb{F}_p$. Further suppose that for every polynomial $P \in \Pi$, and for every pair of terms $t_1, t_2 \in P$, $\deg(t_1 t_2^{p-2}) < d$. Then F has a PC refutation of degree $\max(3pd, d_0)$. Since by our assumption the degree of singular variables is at most d, below we construct a refutation by multiplying each line of the original refutation by non-singular variables to lower their degree to at most $2pd$, which suffices to prove the statement.

For a term t in the proof, let $\mathcal{A}(t)$ denote the subterm consisting of only non-singular variables. Then clearly we have $\mathcal{A}(t)^{p-1} = 1$. Note that we also have $\deg(\mathcal{A}(t_1)^{p-2} t_2) \leq pd$ for any two terms t_1, t_2 by our assumption. For every line P_j in the refutation, we pick a term $t_j \in P_j$ and define $P'_j = \mathcal{A}(t_j)^{p-2} P_j$. Note that $\deg P'_j \leq pd$. We now show that each P'_j can be derived in degree $\max(2pd, d_0)$. If P_j is one of the axioms, we multiply by $\mathcal{A}(t_j)^{p-2}$ to get P'_j, and this takes degree $\max(pd, d_0)$. If $P_j = x_i P_{j_1}$ for $j_1 < j$, we choose t_j such that $t_j = x_i t_{j_1}$. Then we have P'_j is

equal to $x_i P'_{j_1}$ if x_i is singular, and equal to P'_{j_1} otherwise. Finally, let $P_j = P_{j_1} + P_{j_2}$ for $j_1, j_2 < j$. We pick an arbitrary term $t_j \in P_j$. Then we have $P'_j = \mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_1})P'_{j_1} + \mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_2})P'_{j_2}$. We now show that $\deg(\mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_1})) \leq pd$ and $\deg(\mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_2})) \leq pd$ to conclude the proof. Since every term in P_j appears in one of P_{j_1}, P_{j_2}, let $t_j \in P_{j_1}$ without loss of generality. Then we have that t_j, t_{j_1} both appear in P_{j_1} and thus is $\deg(\mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_1})) \leq pd$. If $t_{j_2} \in P_j$ i.e. it is not canceled in the sum $P_{j_1} + P_{j_2}$, then we have t_j, t_{j_2} both appear in P_j and hence $\deg(\mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_2})) \leq pd$. If $t_{j_2} \notin P_j$, this implies that it was canceled in the sum $P_{j_1} + P_{j_2}$ and therefore $t_{j_2} \in P_{j_1}$ and $\deg(\mathcal{A}(t_j)^{p-2}\mathcal{A}(t_{j_2})) \leq d$. □

Lemma 44. *Let Π be a proof and let z be an extension variable such that the corresponding extension axiom implies the line $z^k - 1 = 0$ for some positive integer $k < p$. Let Π' be the proof obtained by reducing each line of Π by $z^k - 1 = 0$. Then we have $|\mathcal{H}_d(\Pi')| \leq |\mathcal{H}_d(\Pi)|$ for any $d \geq 0$.*

Proof. Since the extension axiom for z implies $z^k - 1$, z is nonsingular. Consider a polynomial $P \in \Pi$ and a pair of terms (t_1, t_2) that occur in P. If $Qdeg(t_1, t_2, z) = 0$, then the weight will still be 0 after reducing by $z^k = 1$, and thus $|\mathcal{H}_d(\Pi')| \leq |\mathcal{H}_d(\Pi)|$. □

Lemma 45. *Let $a, b \in \mathbb{F}_p^*$ such that ℓ is the least positive integer less than p with $a^\ell = b^\ell$. Let P be a polynomial in $\mathbb{F}_p[W]$ and let z be an extension variable that occurs in P such that the corresponding extension axiom implies the line $(z-a)(z-b) = 0$. Then, for any two distinct non-negative integers $\ell_1, \ell_0 < \ell$ there exists a unique polynomial $R = R_1 z^{\ell_1} + R_0 z^{\ell_0}$ such that $R = P \mod (z-a)(z-b)$.*

Proof. Since $R = P \mod (z-a)(z-b)$, we have $R(a) = P(a)$ and $R(b) = P(b)$, and therefore it is sufficient to show that there is a unique solution to this pair of equations. Suppose that $\ell_0 < \ell_1$. We have $\begin{vmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{vmatrix} = a^{\ell_0} b^{\ell_0} (a^{\ell_1 - \ell_0} - b^{\ell_1 - \ell_0})$. Since $\ell_1 - \ell_0 < \ell$, this matrix is non-singular over

$\mathbb{F}_p$ and therefore the system of equations

$$R_1 a^{\ell_1} + R_0 a^{\ell_0} = P(a)$$

$$R_1 b^{\ell_1} + R_0 b^{\ell_0} = P(b)$$

has a unique solution. □

Definition 38 ($Split_{z,\ell_1,\ell_0}$)**.** *For a polynomial P and a variable z such that the identity $(z-a)(z-b)=0$ holds, and integers ℓ_1, ℓ_0 such that $\ell_0 < \ell_1$ and $a^{\ell_1-\ell_0} \neq b^{\ell_1-\ell_0}$, let $R = R_1 z^{\ell_1} + R_0 z^{\ell_0}$ be the unique polynomial given by the previous lemma such that $R = P \mod (z-a)(z-b)$. $Split_{z,\ell_1,\ell_0}(P)$ is defined as the pair of polynomials $\{R_1, R_0\}$. For a proof Π, we define $Split_{z,\ell_1,\ell_0}(\Pi)$ to be the set of lines $Split_{z,\ell_1,\ell_0}(P)$ for all $P \in \Pi$.*

Lemma 46. *Let P be a polynomial of the form $P_{\ell-1} z^{\ell-1} + \cdots + P_0$. Then for $\ell_0 < \ell_1 < \ell$, $Split_{z,\ell_1,\ell_0}(P) = \{R_1, R_0\}$ where*

$$R_1 = P_{\ell_1} + \sum_{i<\ell, i\neq\ell_0} c_{1i} P_i$$

$$R_0 = P_{\ell_0} + \sum_{i<\ell, i\neq\ell_1} c_{0i} P_i$$

for some constants $c_{1i}, c_{0i} \in \mathbb{F}_p$.

Proof. This is easily verified from the definition of $Split_{z,\ell_1,\ell_0}$. □

Lemma 47. *Let z be a variable that occurs in a refutation Π but does not occur in any axioms except for $(z-a)(z-b)=0$. Then, for any ℓ_1 and ℓ_0 such that $\ell_0 < \ell_1$ and $a^{\ell_1-\ell_0} \neq b^{\ell_1-\ell_0}$, $Split_{z,\ell_1,\ell_0}(\Pi)$ is a valid refutation and can be derived without increasing the size of $\mathcal{H}_d(\Pi)$ or the singular degree of Π.*

Proof. Let $P_j = P_{j(k-1)} z^{k-1} + \cdots + P_{j2} z^2 + P_{j1} z + P_{j0}$ be the j^{th} line in the refutation Π, where k is the least integer such that the identity $z^k - 1 = 0$ holds (this is without loss of generality

by Lemma 44). We view P_j as a univariate polynomial in z over the appropriate ring. Let $R_j(z) = R_{j1}z^{\ell_1} + R_{j0}z^{\ell_0}$ be a polynomial such that $P_j(z) = R_j(z) \mod (z-a)(z-b)$. Then we have $P_j(a) = R_j(a)$ and $P_j(b) = R_j(b)$, thus by Lemma 45 $R_j(z)$ is uniquely given by

$$\begin{pmatrix} R_{j1} \\ R_{j0} \end{pmatrix} = \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix}^{-1} \begin{pmatrix} P_j(a) \\ P_j(b) \end{pmatrix}$$

We now proceed to show by induction that the set of lines $\{R_{j1}, R_{j0}\}$ is a valid derivation. For the base case, note that all of the axioms are either free of z or eliminated as a result of reducing by $(z-a)(z-b)$, and hence their Split versions are derivable, Now for a line $P_j = \alpha P_{j_1} + \beta P_{j_2}$ for some j_1 and j_2 less than j and $\alpha, \beta \in \mathbb{F}_p$, then we have that $R_{j1} = \alpha R_{j_1 1} + \beta R_{j_2 1}$ and therefore by induction we have a proof of R_{j1} (similarly for R_{j0}). If $P_j = wP_{j'}$ for some $j' < j$ and some variable w distinct from z, we have that $R_{j1} = wR_{j'1}$ (similarly for R_{j0}). Lastly, if $P_j = zP_{j'}$, we have

$$\begin{pmatrix} R_{j'1} \\ R_{j'0} \end{pmatrix} = \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix}^{-1} \begin{pmatrix} P_{j'}(a) \\ P_{j'}(b) \end{pmatrix}$$

from which we need to derive

$$\begin{aligned} \begin{pmatrix} R_{j1} \\ R_{j0} \end{pmatrix} &= \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix}^{-1} \begin{pmatrix} P_j(a) \\ P_j(b) \end{pmatrix} \\ &= \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix}^{-1} \begin{pmatrix} aP_{j'}(a) \\ bP_{j'}(b) \end{pmatrix} \\ &= \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix}^{-1} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} a^{\ell_1} & a^{\ell_0} \\ b^{\ell_1} & b^{\ell_0} \end{pmatrix} \begin{pmatrix} R_{j'1} \\ R_{j'0} \end{pmatrix}. \end{aligned}$$

□

Lemma 48. *Let Π be a proof and let z be an extension variable that does not occur in any of the axioms except the identity $(z-a)(z-b)=0$ for some $a,b\in\mathbb{F}_p^*$ and occurs[1] in at least an ε fraction of pairs (t_1,t_2) in $\mathcal{H}_d(\Pi)$ for an arbitrary integer $d\geq 0$. Then there exist integers $\ell_1,\ell_0<p$ such that the size of $\mathcal{H}_d(Split_{z,\ell_1,\ell_0}(\Pi))$ is at most $(1-\varepsilon/p^2)$ times the size of $\mathcal{H}_d(\Pi)$.*

Proof. The arguments below hold for $\mathcal{H}_d(\Pi)$ for any d, so for simplicity we show the proof for $\mathcal{H}_0(\Pi)=Q(\Pi)$. For a line $P\in\Pi$, let $Q_z(P)$ and $Q_{\neg z}(\Pi)$ be subsets of $Q(P)$ with $Qdeg(t_1,t_2,z)=1$ and $Qdeg(t_1,t_2,z)=0$ for all $(t_1,t_2)\in Q_z(P)$ and $(t_1,t_2)\in Q_{\neg z}(P)$ respectively. By Lemma 44, we assume without loss of generality that Π is reduced by $z^\ell-1=0$ where $\ell>0$ is the least such integer. Let $P=P_{\ell-1}z^{\ell-1}+\cdots+P_2z^2+P_1z+P_0$ and $Q_{i,j}(P)=Q(P_iz^i,P_jz^j)$ for $i,j<\ell$. Then we have

$$Q_z(P)=\sqcup_{i<j}Q_{i,j}(P)$$

$$Q_{\neg z}(P)=\cup_i Q(P_i)$$

where $\sqcup$ denotes disjoint union. This is because by the definition of $Q_{i,j}(P)$, for any pair $(t_1,t_2)\in Q_{i,j}(P)$ we have that $z^i\in t_1$ and $z^j\in t_2$ or vice versa. Therefore, for two pairs (i_1,j_1) and (i_2,j_2) such that $i_1\neq j_1$ and $i_2\neq j_2$ and $\{i_1,j_1\}\neq\{i_2,j_2\}$, we have that $Q_{i_1,j_1}(P)\cap Q_{i_2,j_2}(P)=\emptyset$. Note that this property also extends to $\cup_P Q_{i,j}(P)$. Since $Q(\Pi)=\cup_{P\in\Pi}Q(P)$, we have

$$Q_z(\Pi)=\cup_P\sqcup_{i<j}Q_{i,j}(P)=\sqcup_{i<j}\cup_P Q_{i,j}(P) \tag{3.1}$$

$$Q_{\neg z}(\Pi)=\cup_P\cup_i Q(P_i)$$

and therefore

[1] We say that a variable z occurs in a pair (t_1,t_2) if $Qdeg(t_1,t_2,z)\neq 0$.

$$\mathcal{Q}(\Pi) = \left(\sqcup_{i<j} \cup_P \mathcal{Q}_{ij}(P)\right) \sqcup \left(\cup_P \cup_i \mathcal{Q}(P_i)\right)$$

$$|\mathcal{Q}(\Pi)| = \sum_{i<j} |\cup_P \mathcal{Q}_{ij}(P)| + |\cup_P \cup_i \mathcal{Q}(P_i)| \tag{3.2}$$

Let us now evaluate a similar expression for $\mathcal{Q}(Split_{z,\ell_1,\ell_0}(\Pi))$. Let $\mathcal{Q}^0_{ij}(P) = \mathcal{Q}(P_i, P_j)$. Note that $|\cup_P \mathcal{Q}^0_{ij}(P)| \leq |\cup_P \mathcal{Q}_{ij}(P)|$. Then since by Lemma 46 $Split_{z,\ell_1,\ell_0}(P)$ consists of lines of the form

$$R_1(P) = P_{\ell_1} + \sum_{i<\ell, i\neq\ell_0} c_{1i} P_i$$

$$R_0(P) = P_{\ell_0} + \sum_{i<\ell, i\neq\ell_1} c_{0i} P_i$$

for some constants $c_{1i}, c_{0i} \in \mathbb{F}_p$, we have

$$\begin{aligned}
\mathcal{Q}(Split_{z,\ell_1,\ell_0}(\Pi)) &= \cup_P \mathcal{Q}(Split_{z,\ell_1,\ell_0}(P)) \\
&= \cup_P (\mathcal{Q}(R_1(P)) \cup \mathcal{Q}(R_0(P))) \\
&\subseteq \cup_P \left(\cup_{\ell_0\neq i<j\neq\ell_1} \mathcal{Q}^0_{ij}(P)\right) \cup \left(\cup_i \mathcal{Q}(P_i)\right) \\
&= \left(\cup_{\ell_0\neq i<j\neq\ell_1} \cup_P \mathcal{Q}^0_{ij}(P)\right) \cup \left(\cup_P \cup_i \mathcal{Q}(P_i)\right)
\end{aligned}$$

Therefore, we have that

$$|\mathcal{Q}(Split_{z,\ell_1,\ell_0}(\Pi))| \leq \sum_{\ell_0\neq i<j\neq\ell_1} |\cup_P \mathcal{Q}^0_{ij}(P)| + |\cup_P \cup_i \mathcal{Q}(P_i)| \tag{3.3}$$

$$\leq \sum_{\ell_0\neq i<j\neq\ell_1} |\cup_P \mathcal{Q}_{ij}(P)| + |\cup_P \cup_i \mathcal{Q}(P_i)| \tag{3.4}$$

$$\leq |\mathcal{Q}(\Pi)| - |\cup_P \mathcal{Q}_{\ell_0\ell_1}(P)| \tag{3.5}$$

where the last bound follows from equation 3.2. Now, by our assumption, since $Q_z(\Pi)$ is at least an ε fraction of $Q(\Pi)$ we have from equation 3.1 by an averaging argument that for some $\ell_0 < \ell_1 < \ell$, $\cup_P Q_{\ell_0\ell_1}(P)$ is at least a ε/p^2 fraction of $Q(\Pi)$. For such ℓ_0, ℓ_1 from equation 3.5 we have $|Q(Split_{z,\ell_1,\ell_0}(\Pi))| \leq (1-\varepsilon/p^2)|Q(\Pi)|$.

□

Below we prove a slightly more complex version of the previous lemma.

Lemma 49. *Let Π be a proof, $x \in X$ and let $z = \alpha x + \beta$ be a variable such that x, z do not occur in any other axioms except $x^2 = x$, with z occurring in at least an ε fraction of the pairs in $\mathcal{H}_d(\Pi)$, for some $\alpha, \beta \in \mathbb{F}_p^*$ and any integer $d \geq 0$. Then there exist a refutation Π' such that the size of $\mathcal{H}_d(\Pi')$ is at most $(1-\varepsilon/3p^2)$ times the size of $\mathcal{H}_d(\Pi)$.*

Proof. The proof is by a simple case analysis followed by appealing to the previous lemma. Once again we only show the case of $\mathcal{H}_0(\Pi) = Q(\Pi)$. Let $Q_z(P)$ be the subset of $Q(P)$ with $Qdeg(t_1, t_2, z) = 1$ for all $(t_1, t_2) \in Q_z(P)$. Firstly, note that substituting $x = \alpha^{-1}(z-\beta)$ in the identity $x^2 = x$ we get $(z-a)(z-b) = 0$ for some $a, b \in \mathbb{F}_p$. If either a or b is zero, then we can set $z = 0$ by setting x appropriately to eliminate all terms in $Q(\Pi)$ containing z. Therefore we assume that $a, b \in \mathbb{F}_p^*$, i.e. z is not singular. By Lemma 44, we assume without loss of generality that Π is reduced by $z^\ell - 1 = 0$ where $\ell > 0$ is the least such integer. Then each line P of Π is of the form

$$P = (P'_{\ell-1}z^{\ell-1} + \cdots + P'_2 z^2 + P'_1 z + P'_0) + x(P''_{\ell-1}z^{\ell-1} + \cdots + P''_2 z^2 + P''_1 z + P''_0) + \bar{x}(P'''_{\ell-1}z^{\ell-1} + \cdots + P'''_2 z^2 + P'''_1 z + P'''_0)$$

We define the following subsets of $Q(\Pi)$: (note that since $x \in X$, $Qdeg(x,x,x) = 1$)

$$Q_{zx} = \cup_P \sqcup_{i<j} Q(P'_i z^i, xP''_j z^j) \cup Q(xP''_i z^i, P'_j z^j) \cup Q(xP''_i z^i, xP''_j z^j)$$

$$Q_{z\bar{x}} = \cup_P \sqcup_{i<j} Q(P'_i z^i, \bar{x}P''_j z^j) \cup Q(\bar{x}P''_i z^i, P'_j z^j) \cup Q(\bar{x}P''_i z^i, \bar{x}P''_j z^j)$$

$$Q_{zx\bar{x}} = \cup_P \sqcup_{i<j} Q(xP''_i z^i, \bar{x}P''_j z^j) \cup Q(\bar{x}P''_i z^i, xP''_j z^j)$$

$$Q_z = \cup_P \sqcup_{i<j} Q(P'_i z^i, P'_j z^j)$$

and observe that

$$Q_z(\Pi) = Q_{zx} \sqcup Q_{z\bar{x}} \sqcup Q_{zx\bar{x}} \sqcup Q_z \tag{3.6}$$

Now, if $|Q_{zx} \sqcup Q_{zx\bar{x}}(\Pi)| \geq \varepsilon|Q(\Pi)|/3$, then we set $x = 0$ to obtain a refutation Π_1, for which it is easy to see that

$$Q_z(\Pi_1) = Q_{z\bar{x}} \sqcup Q_z$$

$$Q_{\neg z}(\Pi_1) \subseteq Q_{\neg z}(\Pi)$$

and therefore $|Q(\Pi_1)| \leq (1-\varepsilon/3)|Q(\Pi)|$. Otherwise, if $|Q_{z\bar{x}} \sqcup Q_{zx\bar{x}}(\Pi)| \geq \varepsilon|Q(\Pi)|/3$ then we similarly set $x = 1$ and obtain a refutation Π_1 with $|Q(\Pi_1)| \leq (1-\varepsilon/3)|Q(\Pi)|$.

If both of the above don't hold, then we have $|Q_{zx} \sqcup Q_{z\bar{x}} \sqcup Q_{zx\bar{x}}(\Pi)| \leq 2\varepsilon|Q(\Pi)|/3$ and from equation 3.6 and our assumption we have $|Q_z| \geq \varepsilon|Q(\Pi)|/3$. Let $\ell_0 < \ell_1$ be indices (that exist by an averaging argument) such that $\cup_P Q(P'_{\ell_0} z^{\ell_0}, P'_{\ell_1} z^{\ell_1})$ is of size at least $\varepsilon|Q(\Pi)|/3p^2$. We now substitute $x = \alpha^{-1}(z-\beta)$ in Π and apply $Split_{z,\ell_0,\ell_1}$. It is easy to see that this replaces each line P of Π with two lines of the form

$$R_1(P) = P'_{\ell_1} + \sum_{i<\ell, i\neq\ell_0} c'_{1i} P'_i + \sum_{i<\ell} c''_{1i} P''_i$$

$$R_0(P) = P'_{\ell_0} + \sum_{i<\ell, i\neq\ell_1} c'_{0i} P'_i + \sum_{i<\ell} c''_{1i} P''_i$$

for some constants $c'_{1i}, c'_{0i}, c''_{1i}, c''_{0i} \in \mathbb{F}_p$. By an analysis similar to the previous lemma we

can show that

$$|Q(Split_{z,\ell_0,\ell_1}(\Pi))| \leq |Q(\Pi)| - |\cup_P Q(P'_{\ell_0} z^{\ell_0}, P'_{\ell_1} z^{\ell_1})|$$
$$\leq (1 - \varepsilon/3p^2)|Q(\Pi)|.$$

□

Lemma 50. *Let* Π *be a refutation and let* $Y_i \neq b_1 \dots b_{\log m} \vee E_{b_1 \dots b_{\log m}}$ *be one of its axioms. Then there exists another valid refutation* Π' *with the latter axiom replaced by the axiom* $Y_i \neq b_1 \dots b_{\log m} \equiv \prod_j (y_{ij} - b_j \oplus 1) = 0$*, such that the quadratic degree and singular degree of* Π' *are at most those of* Π*.*

Proof. Note that the axiom $Y_i \neq b_1 \dots b_{\log m} \vee E_{b_1 \dots b_{\log m}}$ can be derived from the axiom $\prod_j (y_{ij} - b_j \oplus 1) = 0$. We construct Π' as follows. We first derive the former axiom from the latter in Π'. Besides this derivation, Π' involves the same steps as Π. Note that since this derivation only involves PCR monomials, this does not raise the quadratic degree of Π'. Also, its singular degree is not more than that of Π. □

3.4.3 Proof of Main Theorem

Theorem 19. *Any* PC *refutation of* $\psi_{N,\kappa,M}(W)$ *is of size at least* $2^{\Omega\left(\frac{n^2}{\kappa^2 2^\kappa (M + n\log(n))}\right)}$.

Proof. Let Π be a refutation of $\psi_{N,\kappa,c}(W)$ of size at most $2^{\gamma n^2/(\kappa^2 2^\kappa (M + n\log(n)))}$ for a small enough constant γ. Given an alleged PC refutation Π, Algorithm 1 (defined below) will apply a sequence of restrictions and cleanup steps in order to produce a refutation Π'' of a restricted version of $\psi_{N,\kappa,c}(W)$ with the property that both the singular as well as the quadratic degree of Π'' are at most d. The algorithm contains two while loops, the first of which iteratively removes all terms of high singular degree, and the second iteratively removes all pairs of terms of high quadratic

Input: A refutation Π of $\psi_{N,\kappa,M}$
Output: A refutation Π' with quadratic and singular degree less than d

$d \leftarrow \nu n/\kappa$, where ν is a sufficiently small constant.
$M' \leftarrow M + n\log(n)$.
$S \leftarrow$ the set of all terms in the proof of singular degree greater than d
$B \leftarrow \emptyset$.
while *S is non empty* **do**
    Pick a variable w that, by an averaging argument, occurs in at least d/M' fraction of terms in S
    if $w \in X \cup Y$ **then**
        Substitute $w = 0$
        Call **X-cleanup**$(w = 0)$ or **Y-cleanup**$(w = 0)$ depending on whether $w \in X$ or $w \in Y$
    end
    if *w is an extension variable, defined by $w = f(X,Y)$* **then**
        Let $\sigma = \sigma_X \cup \sigma_Y$ be an assignment to the variables of f such that $f(\sigma) = 0$
        Substitute σ
        Call **X-cleanup**(σ_X) and **Y-cleanup**(σ_Y)
    end
end
$H \leftarrow$ the set of all pairs of terms in the proof of quadratic degree greater than d
while *H is non empty* **do**
    Pick a variable w that, by an averaging argument, occurs in at least d/M' fraction of terms in H
    if $w \in X \cup Y$ **then**
        Substitute $w = 0$
        Call **X-cleanup**$(w = 0)$ or **Y-cleanup**$(w = 0)$ depending on whether $w \in X$ or $w \in Y$
    end
    if *w is an extension variable, defined by $w = f(X,Y)$* **then**
        Let $\sigma = \sigma_X \cup \sigma_Y$ be an assignment to the variables of f such that $f(\sigma) = \alpha x + \beta$ for some $x \in X$ (exists since we substitute a complete assignment for Y eventually and hence extension variables that depend only on Y are inconsequential)
        Substitute σ
        Call **X-cleanup**$(\sigma_X \cup \{x = 0\})$ and **Y-cleanup**(σ_Y)
        Split on w using Lemma 49
    end
end

Algorithm 3: Eliminating high quadratic and singular degree terms from the proof

degree. From Π'', we will apply a further restriction to all of the remaining unset Y variables, to extract a refutation of a subset of m' equations from $F_{n,k}$ of low degree, contradicting the degree lower bound given in Lemma 41. Recall that $F_{n,k}$ is defined over variables X and we pick a subset of these equations by matching pigeons Y_i to equations in $F_{n,k}$ through a complete bipartite graph. We initialize a bad list B of bit strings $b_1 \dots b_{\log m}$ to empty (where each such bit string indexes an equation $E_{b_1 \dots b_{\log m}}$ of $F_{n,k}$). This bad list will contain all of the equations that were affected by either of the above while loops.

We will first analyze the first while loop (lines 5-15). Initially S is initialized to the set of all terms in the proof of singular degree greater than d. Let $M' = M + n\log(n)$. This loop kills off terms in S until S is empty, by iteratively picking a variable w that, by an averaging argument, occurs in at least a d/M' fraction of terms in S. There are two cases depending on whether $w \in X \cup Y$ (the first case) or whether $w \in Z$. In the first case, we apply the restriction $w = 0$ and call **X-cleanup**$(w = 0)$ or **Y-cleanup**$(w = 0)$ depending on whether $w \in X$ or $w \in Y$. This eliminates the contribution to high singular degree from terms containing w, and hence obtains a $(1 - d/M')$-factor reduction in the size of S. In the second case, w is an extension variable, defined by $w = f(X,Y)$ for some polynomial f that depends on at most κ variables from $X \cup Y$. Since w is singular, there exists an assignment $\sigma = \sigma_X \cup \sigma_Y$ such that $f(\sigma) = 0$. We apply the restriction σ to the proof, thus eliminating all terms containing w, which causes a $(1 - d/M')$-factor reduction in the number of high singular degree terms. Next, we run subroutines **X-cleanup**(σ_X) and **Y-cleanup**(σ_Y) (described below) to get rid of all axioms that were affected by the restriction σ. without affecting the other axioms. By repeating the above for $-\log|S|/\log(1 - d/M')$ $\approx M' \log|S|/d \leq O(\gamma)n/\kappa 2^\kappa$ iterations (where $|S| = 2^{\gamma n^2/\kappa^2 2^\kappa M'}$), we eliminate all terms in S from the proof and thus obtain a refutation of singular degree less than d. The processes **X-cleanup**(σ_X) and **Y-cleanup**(σ_Y) increase the size of the bad list B by only $O(|\sigma|) = O(\kappa)$ per call, since each X-variable occurs in at most $k_1 = O(1)$ clauses, and each clause has size $k = O(1)$). Therefore the total size of B at the end of the first while loop is at most $O(\gamma)n/2^\kappa$.

Let Π' be the (modified) proof after exiting the first while loop. Before entering the second while loop (lines 18-29), we initialize H to be equal to $\mathcal{H}_d(\Pi')$, the set of all pairs of terms of Π' of quadratic degree greater than d. Note that H may be different than the original set of bad pairs, since during the execution of the first while loop, some extension variables that were originally singular may become nonsingular. In this second loop, we will kill off all pairs from H by iteratively picking a variable w that contributes to the weight of at least a d/M' fraction of pairs in H.

There are two cases depending on whether $w \in X \cup Y$ or $w \in Z$. In the first case ($w \in X \cup Y$), we apply the restriction $w = 0$ and call **X-cleanup**$(w = 0)$ or **Y-cleanup**$(w = 0)$ depending on whether $w \in X$ or $w \in Y$ respectively. This eliminates the contribution to high quadratic degree from terms containing w, and hence obtains a $(1 - d/M')$-factor reduction in the size of H. In the second case, w is an extension variable defined by extension axiom $w = f(X,Y)$ where f depends on at most κ variables from $X \cup Y$. We can assume that z depends on at least one X-variable since at the end of the procedure we will set all Y-variables to constants, and therefore extension variables that only depend on Y variables will be inconsequential. Thus, there there exists an assignment $\sigma = \sigma_X \cup \sigma_Y$ such that $f(\sigma) = \alpha x + \beta$ for some $\alpha, \beta \in \mathbb{F}_p$ and $x \in X$. We apply σ to the proof, and then call the subroutines **X-cleanup**$(\sigma_X \cup \{x = 0\})$ and **Y-cleanup**(σ_Y) to get rid of all axioms that were affected by σ and also those that contain x. Now that the axioms are free of x and w, we Split on w using Lemma 49, which causes a $(1 - d/3p^2M')$-factor reduction in the number of high quadratic degree terms. By repeating the above for $-\log|H|/\log(1 - d/3p^2M')$ $\approx p^2M' \log|H|/d \leq O(\gamma)n/\kappa 2^\kappa$ iterations (where $|H| = 2^{2\gamma n^2/\kappa^2 2^\kappa M'}$), we eliminate all terms in H from the proof and thus obtain a refutation of quadratic degree less than d. Since one call to **X-cleanup**(σ_X) and **Y-cleanup**(σ_Y) increases the size of the bad list B by only $O(|\sigma|) = O(\kappa)$ per call, the total size of B upon termination of Algorithm 1 is at most $O(\gamma)n/2^\kappa$.

Let Π'' denote the modified proof upon termination of Algorithm 1. We claim that the singular degree as well as the quadratic degree of Π'' is at most d. This is because the first

while loop gets rid of all terms of high singular degree, and neither the first or second subroutine creates new variables of high singular degree since substitution can only turn singular variables to nonsingular and not the other way around. Similarly, the second while loop gets rid of all pairs of terms of high quadratic degree, and this second while loop does not create new terms of high quadratic degree, since by Observation 1 substitution does not increase the quadratic degree.

Note that out of the $m' = (1-\varepsilon/2)m$ pigeons, there are at least a $m' - O(\gamma)n/2^{\kappa}$ pigeons still alive (i.e. not removed by the operations **Y-cleanup**), since we run for $O(\gamma)n/\kappa 2^{\kappa}$ many iterations, and in each iteration at most κ pigeons are affected by any extension variable. Since $|B| \leq O(\gamma)n/2^{\kappa}$, the number of untouched equations available for the pigeons to map to is $m - O(\gamma)n/2^{\kappa}$. We now substitute for the remaining pigeons Y_i so that we select a subset of at least $(1-\varepsilon)m$ unsatisfiable equations φ not in B from $F_{n,k}$ and obtain a refutation of them of quadratic degree at most d and singular degree at most d (assuming γ is small enough). By Lemma 43, this implies a refutation of φ of degree at most $O(p^2 d)$. Now, for all surviving extension variables we substitute them with their definitions in terms of the variables X. Note that since the extension variables are degree κ polynomials this raises the degree to at most $O(p^2 \kappa d)$. Since $d = \nu n/\kappa$, for sufficiently small ν, we end up with a refutation of φ of degree less than $c_2 n$, contradicting Lemma 41.

For the cleanup operations to work properly, recall that $|B| = O(\gamma)n/2^{\kappa}$ always holds conditioned on each cleanup operation increasing B only by $O(\kappa)$ (for a small enough constant γ).

Input: A partial assignment σ to the variables X

for *Every* $x \in X$ *that occurs in* σ **do**
  for *Every equation* $E_{b_1 \dots b_{\log m}}$ *from* $F_{n,k}$ *that* x *occurs in* **do**
    Add $E_{b_1 \dots b_{\log m}}$ to the list B
    for *Every* $i \in [m']$ **do**
      Use Lemma 50 to replace the axiom $Y_i \neq b_1 \dots b_{\log m} \vee E_{b_1 \dots b_{\log m}}$ by $Y_i \neq b_1 \dots b_{\log m} \equiv \prod_j (y_{ij} - b_j \oplus 1) = 0$
    end
  end
end

Algorithm 4: X-cleanup

X-cleanup(σ) Correctness. Suppose $x \in X$ is a variable that occurs in σ. We first add all the equations in $F_{n,k}$ that x occurs in to the list B. By Lemma 41 this is k_1 many equations. We now proceed to eliminate all axioms that contain x. For every such equation $E_{b_1 \dots b_{\log m}}$ from $F_{n,k}$, which appears in the axiom $Y_i \neq b_1 \dots b_{\log m} \vee E_{b_1 \dots b_{\log m}}$ for every i, we use Lemma 50 to replace the latter by $Y_i \neq b_1 \dots b_{\log m} \equiv \prod_j (y_{ij} - b_j \oplus 1) = 0$ for every i. That is, we assert that no pigeon maps to the equation $E_{b_1 \dots b_{\log m}}$ and hence it stands eliminated. By Lemma 50 this does not raise the quadratic/singular degree of the proof. We do this for all such x. Note that we have maintained the property that one call to this process adds $O(|\sigma|)$ entries to the bad list B.

Y-cleanup(σ) Correctness. Let i be an index such that $y_{ij} \in Y_i$ is a variable that appears in σ for some j. For each such index i, our plan is to map the i^{th} pigeon comprising of variables $y_{i1} \cdots y_{i \log m}$ to some equation $E_{b_1 \cdots b_{\log m}}$ that is not on the bad list B, and then satisfy the latter. Firstly, note that only κ variables from Y_i can appear in σ (since extension variables that **Y-cleanup** is called on depend on only κ variables and hence the size of σ is bounded by κ). Therefore, there are at least $m/2^{\kappa}$ values that the binary string $y_{i1} \cdots y_{i \log m}$ can be set to, given that some

Input: A partial assignment σ to the variables Y

for *Each i such that $y_{ij} \in Y_i$ is a variable that appears in σ for some j* **do**

  Pick an assignment $b_1 \dots b_{\log m}$ to $y_{i1} \cdots y_{i\log m}$ such that the variables that appear in σ are set consistently, and $b_1 \dots b_{\log m}$ does not appear in the bad list B (this is possible since the size of B is small enough; see paragraph below)

  Apply $b_1 \dots b_{\log m}$ to $y_{i1} \cdots y_{i\log m}$; this turns the axiom $Y_i \neq b_1 \cdots b_{\log m} \vee E_{b_1 \cdots b_{\log m}}$ into $E_{b_1 \cdots b_{\log m}} = 0$

  Let σ_X be a partial assignment to the X-variables such that σ_X satisfies $E_{b_1 \cdots b_{\log m}}$

  Substitute σ_X and add any equation that contains a variable from σ_X to the bad list B

  Add $b_1 \dots b_{\log m}$ to the bad list B

end

Algorithm 5: Y-cleanup

of these variables are already set by σ. Since the size of the set B is always $O(\gamma)n/2^{\kappa}$, there exists an assignment $b_1 \dots b_{\log m}$ to $y_{i1} \cdots y_{i\log m}$ such that the variables that appear in σ are set consistently, and $b_1 \dots b_{\log m}$ does not appear in the bad list B. We apply the assignment $b_1 \dots b_{\log m}$ to $y_{i1} \cdots y_{i\log m}$. Note that this turns the axiom $Y_i \neq b_1 \cdots b_{\log m} \vee E_{b_1 \cdots b_{\log m}}$ into $E_{b_1 \cdots b_{\log m}} = 0$ (i.e. selects the equation $E_{b_1 \cdots b_{\log m}}$). We now get rid of it as follows. Since $b_1 \cdots b_{\log m}$ is not on the bad list B, the equation $E_{b_1 \cdots b_{\log m}}$ is untouched, i.e. none of its variables have been set before. Let σ_X be a partial assignment to the X-variables such that σ_X satisfies $E_{b_1 \cdots b_{\log m}}$. We substitute σ_X and add any equation that contains a variable from σ_X to the bad list B. Finally, we add $b_1 \dots b_{\log m}$ to the bad list B. Note that we have maintained the property that one call to this process adds only a $O(|\sigma|)$ number of entries to the bad list B since the equations $F_{n,k}$ contain only k variables per equation.

□

Acknowledgments

The authors would like to thank Paul Beame and Dmitry Sokolov for helpful discussions. Chapter 3 contains material from "Lower bounds for Polynomial Calculus with extension

variables over finite fields." by Russell Impagliazzo, Sasank Mouli, and Toniann Pitassi, which is currently published on the Electronic Colloquium on Computational Complexity and being prepared for a formal publication.

Chapter 4

Open Problems

The obvious open problem is to prove a lower bound for $\mathsf{AC}^0[p]$-Frege systems, whether using algebraic proofs or not. As stepping stones towards this goal, we think it would be interesting to answer the following:

1. Prove lower bounds for the system Trinomial-ΠΣ-PC.

2. Our simulations from Chapter 2 require a sufficiently large extension field. Can we either p-simulate polynomial calculus over a large extension field with polynomial calculus over the base field, or prove that no simulation exists?

3. How far can our lower bounds from the previous chapter be extended?

Bibliography

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole principle. *Combinatorica*, 14(4):417–433, 1994.

[Ale21] Yaroslav Alekseev. A lower bound for polynomial calculus with extension rule. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[All89] Eric Allender. A note on the power of threshold circuits. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 580–584. IEEE, 1989.

[AR01] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 190–199. IEEE Computer Society, 2001.

[BC96] Samuel R Buss and Peter Clote. Cutting planes, connectivity, and threshold logic. *Archive for Mathematical Logic*, 35(1):33–62, 1996.

[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.

[BIK+92] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, pages 200–220, 1992.

[BIK+94] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bounds on hilbert's nullstellensatz and propositional proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 794–806. IEEE, 1994.

[BIK+97] Samuel R. Buss, Russell Impagliazzo, Jan Krajícek, Pavel Pudlák, Alexander A. Razborov, and Jirí Sgall. Proof complexity in algebraic systems and bounded depth

frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1997.

[BKPS02] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and davis–putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002.

[BKZ15] Samuel Buss, Leszek Kołodziejczyk, and Konrad Zdanowski. Collapsing modular counting in bounded arithmetic and constant depth propositional proofs. *Transactions of the American Mathematical Society*, 367(11):7517–7563, 2015.

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.

[BSW99] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 517–526, 1999.

[CCT87] William Cook, Collette R Coullard, and Gy Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 174–183, 1996.

[CK13] Peter Clote and Evangelos Kranakis. *Boolean functions and computation models*. Springer Science & Business Media, 2013.

[CR79] Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. *The journal of symbolic logic*, 44(1):36–50, 1979.

[CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM (JACM)*, 35(4):759–768, 1988.

[GH03] Dima Grigoriev and Edward A Hirsch. Algebraic proof systems over formulas. *Theoretical Computer Science*, 303(1):83–102, 2003.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates VS. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[GL10] Nicola Galesi and Massimo Lauria. Optimality of size-degree tradeoffs for polynomial calculus. *ACM Trans. Comput. Log.*, 12(1):4:1–4:22, 2010.

[GV01] Dima Grigoriev and Nicolai Vorobjov. Complexity of null-and positivstellensatz proofs. *Annals of Pure and Applied Logic*, 113(1-3):153–160, 2001.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical computer science*, 39:297–308, 1985.

[IMP20] Russell Impagliazzo, Sasank Mouli, and Toniann Pitassi. The surprising power of constant depth algebraic proofs. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 591–603, 2020.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiri Sgall. Lower bounds for the polynomial calculus and the gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

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

[Mar62] Davis Martin. A machine program for theorem-proving. *Communications of the ACM*, 5(7):397, 1962.

[MN15] Mladen Miksa and Jakob Nordström. A generalized method for proving polynomial calculus degree lower bounds. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 467–487. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[MP98] Alexis Maciel and Toniann Pitassi. Towards lower bounds for bounded-depth frege proofs with modular connectives. *Proof complexity and feasible arithmetics*, 39:195–227, 1998.

[MT98] Alexis Maciel and Denis Thérien. Threshold circuits of small majority-depth. *Inf. Comput.*, 146(1):55–83, 1998.

[Pit96] Toniann Pitassi. Algebraic propositional proof systems. In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996*, pages 215–244, 1996.

[Raz87] Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

[Raz98a] Alexander A Razborov. Lower bounds for the polynomial calculus. *computational complexity*, 7(4):291–324, 1998.

[Raz98b] Alexander A Razborov. Lower bounds for the polynomial calculus. *computational complexity*, 7(4):291–324, 1998.

[RT08a] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

[RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *computational complexity*, 17(3):407–457, 2008.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[Sok20] Dmitry Sokolov. (semi) algebraic proofs over $\{\pm 1\}$ variables. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 78–90, 2020.