Electronic Colloquium on Computational Complexity, Report No. 24 (2019)

The Surprising Power of Constant Depth Algebraic Proofs

Russell Impagliazzo
russell@cs.ucsd.edu

Sasank Mouli
galivenk@ucsd.edu

Toniann Pitassi
toni@cs.toronto.edu

Abstract

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 ([22, 24]), 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 ([2]), Polynomial Calculus ([8]), and SOS ([11]). However to date there has been no progress on $\mathsf{AC}^0[p]$-Frege lower bounds.

In this paper we study constant-depth extensions of the Polynomial Calculus [10]. We show that these extensions are much more powerful than was previously known. Our main result is that small depth Polynomial Calculus (over a sufficiently large field) can polynomially simulate all of the well-studied semialgebraic proof systems: Cutting Planes, Sherali-Adams, Sum-of-Squares (SOS), and Dynamic SOS. Additionally, they can also quasi-polynomially simulate $\mathsf{AC}^0[p]$-Frege as well as TC^0-Frege. Thus, proving strong lower bounds for $\mathsf{AC}^0[p]$-Frege would seem to require proving lower bounds for systems as strong as TC^0-Frege.

ISSN 1433-8092

1 Introduction

Proof complexity has evolved in parallel to circuit complexity, typically with circuit lower bound techniques being eventually used to show lower bounds for analogous proof systems. One stubborn exception is the analogous proof system for $\mathsf{AC}^0[p]$, the class of bounded depth circuits with prime modular counting gates. Despite strong lower bounds for this class dating back thirty years ([22, 24]), there are no significant lower bounds for $\mathsf{AC}^0[p]$-Frege. Algebraic proof systems (Nullstellensatz ([2]), polynomial calculus ([8]), Positivestellensatz aka sum of squares ([11]), ideal proofs ([12])) were invented to try to transport some of the algebraic reasoning used by Razborov and Smolensky to the proof complexity case, but while interesting in their own right, lower bounds for these systems have not been translated into lower bounds for $\mathsf{AC}^0[p]$-Frege. This paper offers one explanation for this failure: small modifications of these algebraic proof systems to handle constant depth overshoot and allow reasoning far beyond that possible by $\mathsf{AC}^0[p]$ circuits.

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 ([18, 10, 7]). 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 [23], [15] and Tseitin tautologies [4]. 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 [10] 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 [10] 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 [21] 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.

1.1 Our Work

Here, 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 $a_i, b \in F$. 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 $d-2$ layers of additive and multiplicative extensions, with the layers alternating between additive and multiplicative extensions. (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.)

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 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.* [5] 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 [10], 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 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 Dynamic Sum of Squares.

1.2 Related Work

Pitassi [18, 19] introduced powerful generalizations of the Polynomial Calculus that operate directly on formulas. Grochow and Pitassi [13] introduced the more general IPS proof system, and proved that superpolynomial lower bounds for IPS would imply the longstanding problem of separating VP from VNP. However, these algebraic systems are not Cook-Reckhow proof systems since proofs are not known to be checkable in polynomial time (but rather in randomized polynomial-time.)

In 2003, Grigoriev and Hirsch [10] introduced a Cook-Reckhow style algebraic proof system for formulas, with derivation rules corresponding to the ring axioms. Motivated by understanding how many basic ring identities are needed to verify polynomial identities, Hrubes and Tzameret [14] introduced a very closely related equational proof system for proving polynomial identities over a ring. Even earlier, [7] study essentially the same proof system but where the focus is over finite fields. Finally, Raz and Tzameret [20] introduced the *Res*(*lin*) proof system, which generalizing Resolution using extension variables, in a similar way to our generalization of PC using extension variables.

In this paper, we study the same system introduced by Grigoriev and Hirsch – proofs operate with algebraic formulas, and with explicit rules for manipulating equivalent polynomials. Over sufficiently large fields, we show that constant-depth proofs are much more powerful than what was previously thought. Our main result is that low depth algebraic proofs can polynomially simulate all of the well-studied semialgebraic proof systems: Cutting Planes, Sherali-Adams and Sum-of-Squares proofs, and they can also simulate constant-depth Frege proofs with mod gates and with threshold gates ($\mathsf{AC}^0[p]$-Frege and TC^0-Frege).

The rest of the paper is organized as follows. In section 2.1, we discuss some basic definitions. In section 2.2, we formalize the notion of transformations. In section 4.2, we formally state and prove Theorem 1 for Cutting Planes with bounded coefficients. In section 4.3, we extend the simulation to the semantic case. In section 4.4, we prove an analog of the results in section 4.2 over a large enough finite field extension. In sections 5 to 6, we use this analog to prove Theorems 2 and 3. Finally in section 7, we prove Theorem 4.

2 Preliminaries and Generalizations of Polynomial Calculus

2.1 Preliminaries

2.1.1 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 objects as scalars. 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 1. *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.

Definition 2. *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* α, $\beta \in \mathbb{F}$

$R_\ell = x_i R_k$ *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 Generalizations of Polynomial Calculus

We now define a variant of Polynomial Calculus where the proof system is additionally allowed to introduce new variables y_j corresponding to affine forms in the original variables x_i.

Definition 3. $\Sigma\Pi\Sigma$*-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 $\Sigma\Pi\Sigma$-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 $\Sigma\Pi\Sigma$-PC refutation is equal to the size of the Polynomial Calculus refutation of Γ'.

Though $\Sigma\Pi\Sigma$-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 4. *A Trinomial is a polynomial with at most three monomials*

Definition 5. *Trinomial-$\Pi\Sigma$-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.

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

Definition 6. *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'

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 $x_1 y_1$.

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.

3 Formal statement of results and Proof sketch

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

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

Theorem 1 is proved in sections 4.2 and 4.3.

Theorem 2. *There is a fixed constant d such that Depth-d-PC over $\mathbb{F}_{p^m}$ can quasipolynomially simulate $\mathsf{AC}^0[q]$-Frege for any prime q.*

We prove Theorem 2 in sections 5.1 and 5.2.

Theorem 3. *A TC^0-Frege proof of depth d can be p-simulated by Depth-d'-PC where $d' = O(d)$.*

The proof of Theorem 3 is shown in section 6.

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.*

Theorem 4 is proved in section 7.

3.1 Key ideas and proof sketch

In this section we outline how we translate inequalities into polynomials, and simulate proofs involving these inequalities into polynomial calculus derivations over their translations. Here, we consider the somewhat simpler case when the underlying field is $\mathbb{Q}$ or any other characteristic zero field. In section 4.4, we do the analogous translation for any sufficiently large finite field.

Let us start by considering 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 7. *Translation from CP* to Trinomial-$\Pi\Sigma$-PC*
For a line $A_j \equiv \sum_i a_{ij}x_i \geq b_j$ its translation in Trinomial-$\Pi\Sigma$-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 Trinomial-$\Pi\Sigma$-PC the equation $z \in A \cap B$. We call this the Intersection lemma. We prove this lemma formally in the next section.

Simulating syntactic CP* In section 4.2, we show how all the derivations rules of syntactic CP* can be simulated with the help of the Intersection lemma, hence proving Theorem 1. 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 $z' = z/c$.

Simulating semantic CP* In section 4.3, we derive a semantic consequence $y_3 \in C$ of two lines $y_1 \in A$ and $y_2 \in B$ by simulating a dynamic programming based approach to derive an equation for every infeasible value of the tuple (y_1, y_2, z) in the grid $I_1 \times I_2 \times I_3$, where I_j is all the integer values y_j can take. We then use the Intersection lemma again to narrow down the range of values for y_3 based on the constraints $y_1 \in A$ and $y_2 \in B$.

Simulating in $\mathbb{F}_p^m$ In section 4.4, we show the simulation of syntactic CP* over a finite field extension $\mathbb{F}_p^m$. To represent large numbers over a finite field we use the linear transformation $x_i \mapsto \alpha^{x_i}$ for every boolean variable x_i, where α is a primitive element of $\mathbb{F}_p^m$. The simulation is then largely similar to section 4.2.

Simulating $\mathsf{AC}^0[q]$-Frege and TC^0-Frege Looking at the simulations so far, it is already clear that Depth-d-PC can express ORs of statements of the form $z \in A$. In sections 5.1, 5.2 and 6 we show that Depth-d-PC is able to simulate a form of sequent calculus involving threshold and modular connectives introduced in [6] and [16], from which Theorems 2 and 3 follow.

Dealing with large coefficients In section 7, we show how large integers can be represented as bit vectors in our system, define addition and multiplication operations over these vectors and show that our system can prove some basic properties of them. This is sufficient for us to complete simulations of Cutting Planes with unbounded coefficients and Dynamic Sum of Squares.

4 Small-weight Cutting Planes Simulations

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 inequalities $B_1 \cdots B_s$ 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*

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

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 length of a Cutting Planes proof is equal to the number of inequalities 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. **CP*** *is a subsystem of Cutting Planes where the absolute value of each coefficient is bounded by a polynomial in the number of variables. Without loss of generality, the sizes of the coefficients can be bounded by $2^{O(n^2)}$ due to [**?**].*

4.1 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 1. *"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 2. *Let $Q(z-a)=0$ and $Q\prod_{i=1}^{k}(z-b_i)=0$ be two equations in Trinomial-$\Pi\Sigma$-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

$$Qz_1=0 \tag{1}$$

$$QQ_1z_2=0 \tag{2}$$

$$z_1=z-a \tag{3}$$

$$z_2=z-b \tag{4}$$

Multiplying equation (1) by Q_1, we have

$$QQ_1z_1=0 \tag{5}$$

Let $c=a-b$. By subtracting (4) from (3) we derive

$$z_1-z_2+c=0 \tag{6}$$

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 (5) and $QQ_1z_2=0$ from (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 3. *"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

$$\prod_{a \in A} (z - a) = 0 \tag{7}$$

$$\prod_{b \in B} (z - b) = 0 \tag{8}$$

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 (7) can be written as

$$Q_1(z - a_1) = 0 \tag{9}$$

Multiplying (8) by Q_2 we get

$$\prod_{b \in B \cup A_2} (z - b) = 0 \tag{10}$$

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

$$\prod_{b \in A_1 \cup B_0} (z - b) = 0 \tag{11}$$

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

$$Q_1 \prod_{b \in B_0} (z - b) = 0 \tag{12}$$

Now since $a_1 \notin B_0$, we can apply Lemma 2 on equations (9) and (12) to get

$$Q_1 = 0$$

i.e.

$$\prod_{a \in A_1} (z - a) = 0 \tag{13}$$

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

Now we have two sets of constants A_1 and B with corresponding equations (13) and (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.

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

We are now ready to state and prove our simulation theorem for syntactic CP*

Theorem 4. *Trinomial-ΠΣ-PC over $\mathbb{Q}$ can simulate syntactic Cutting Planes with the number of lines polynomial in n and the coefficient size.*

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 4. *Simulating addition*
Let $x(x-1)\cdots(x-a) = 0$ and $y(y-1)\cdots(y-b) = 0$ be two equations in a Trinomial-$\Pi\Sigma$-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

$$z_j Y_j - x_0 Y_j = 0 \tag{14}$$

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

$$z_j Y_j - z_{j+i} Y_j - i Y_j = 0 \tag{15}$$

Subtracting this from (14) we get

$$z_{j+i} Y_j - x_0 Y_j + i Y_j = 0 \tag{16}$$

By the definition of x_i we have

$$x_i = x_0 - i$$

Multiplying the above equation by the monomial Y_j, we get

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

Subtracting the above equation from (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

$$z_j \cdots z_{j+i} Y_j - x_0 \cdots x_{i-1} z_{j+i} Y_j = 0 \tag{17}$$

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

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

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

$$x_0 \cdots x_{i-1}z_{j+i}Y_j - x_0 \cdots x_iY_j = 0 \tag{18}$$

Subtracting (18) from (17) we get

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

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

$$z_j \cdots z_{j+a}Y_j - x_0 \cdots x_aY_j = 0 \tag{19}$$

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 $\{z_k : 0 \le k < j\} \cup \{z_k : j + a < k \le a + b\}$, 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 $O(ab + b^2)$. $\square$

Corollary 1. *Given the translations of $\sum_i a_{ij}x_i \ge b_j$ and $\sum_i a_{ik}x_i \ge b_k$, we can derive in Trinomial-$\Pi\Sigma$-PC the translation of $\sum_i (a_{ik} + a_{ij})x_i \ge b_j + b_k$ 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 $y = \sum_i a_{ik}x_i - b_k$, $b = \sum_i a^+_{ik} - b_k$. $\square$

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

Lemma 5. *Let $(z - a_1) \cdots (z - a_k) = 0$ be an equation in Trinomial-$\Pi\Sigma$-PC. We can derive the equation*

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

where $z' = cz$ in Trinomial-$\Pi\Sigma$-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-$\Pi\Sigma$-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 4

Corollary 3. *Let $z = \sum_i a_{ij} x_i$ be an equation in Trinomial-$\Pi\Sigma$-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 4 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 6. *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.

$$z\Big(z-1\Big)\cdots\Big(z-(\sum_i a_{ij}^+)\Big)=0 \tag{20}$$

Now, using Lemma 5 on the equation $(cz-b)(cz-(b+1))\cdots(cz-d)=0$ with the multiplication constant equal to $1/c$, we can derive

$$z(z-b/c)\cdots(z-d/c)=0 \tag{21}$$

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 (20) and (21), we see that only the integer values are retained from (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-$\Pi\Sigma$-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.

4.3 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 $L_1 \equiv \sum_i a_ix_i \geq d_1$, $L_2 \equiv \sum_i b_ix_i \geq d_2$ be two lines in a Cutting Planes proof and let $L_3 \equiv \sum_i c_ix_i \geq d_3$ be a semantic consequence of L_1 and L_2. Let $y=\sum_i a_ix_i$, $z=\sum_i b_ix_i$ and $w=\sum_i c_ix_i$. 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 7. *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 $y_\ell = \sum_{i=1}^{\ell} a_i x_i$ 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

$$y_1 = a_1 x_1$$
$$z_1 = b_1 x_1$$

derive

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

and multiply by y_1 to derive

$$y_1(z_1 - b_1) = a_1 b_1 x_1(x_1 - 1) = 0$$

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 4)

$$(x_\ell - 1) \prod_{\substack{a \in A_{\ell-1} \\ a \neq i}} (y_\ell - a) \prod_{\substack{b \in B_{\ell-1} \\ b \neq j}} (z_\ell - b) \prod_{\substack{c \in C_{\ell-1} \\ c \neq k}} (w_\ell - c) \tag{22}$$

$$= (x_\ell - 1) \prod_{\substack{a \in A_{\ell-1} \\ a \neq i}} (y_{\ell-1} - a) \prod_{\substack{b \in B_{\ell-1} \\ b \neq j}} (z_{\ell-1} - b) \prod_{\substack{c \in C_{\ell-1} \\ c \neq k}} (w_{\ell-1} - c) \tag{23}$$

$$= 0 \tag{24}$$

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

$$x_\ell \prod_{\substack{a\in A_{\ell-1}\\ a\neq i-a_\ell}} (y_\ell - (a+a_\ell)) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j-b_\ell}} (z_\ell - (b+b_\ell)) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k-c_\ell}} (w_\ell - (c+c_\ell)) \tag{25}$$
$$= x_\ell \prod_{\substack{a\in A_{\ell-1}\\ a\neq i-a_\ell}} (y_{\ell-1} - a) \prod_{\substack{b\in B_{\ell-1}\\ b\neq j-b_\ell}} (z_{\ell-1} - b) \prod_{\substack{c\in C_{\ell-1}\\ c\neq k-c_\ell}} (w_{\ell-1} - c) \tag{26}$$
$$= 0 \tag{27}$$

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 (22) and (25)

$$(x_\ell - 1) \prod_{\substack{a\in A_\ell\\ a\neq i}} (y_\ell - a) \prod_{\substack{b\in B_\ell\\ b\neq j}} (z_\ell - b) \prod_{\substack{c\in C_\ell\\ c\neq k}} (w_\ell - c) = 0 \tag{28}$$

$$x_\ell \prod_{\substack{a\in A_\ell\\ a\neq i}} (y_\ell - a) \prod_{\substack{b\in B_\ell\\ b\neq j}} (z_\ell - b) \prod_{\substack{c\in C_\ell\\ c\neq k}} (w_\ell - c) = 0 \tag{29}$$

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-$\Pi\Sigma$-PC, for $i = 1, 2, 3$. Let $\mathcal{I}_{a,b}$ 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 $I \in \mathcal{I}_{a,b}$, $\mathcal{R}_w(I)$ gives a list of possible values the variable w can take when $(y, z) = (a, b)$. By Lemma 7, $(y, z, w) = (a, b, c)$ is infeasible if and only if there is an equation $I \in \mathcal{I}_{a,b}$ such that $c \notin \mathcal{R}_w(I)$. Therefore, $\bigcap_{I\in\mathcal{I}_{a,b}} \mathcal{R}_w(I)$ 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-$\Pi\Sigma$-PC as follows. To simulate the inner **for** loop, we use the Intersection lemma inductively over all equations in $\mathcal{I}_{a,b}$ to get equations $J_{a,b}$ such that

```
procedure $w$-FEASIBLE($P_1$,$P_2$)
    $C' \leftarrow \emptyset$
    for $(a, b) \in \mathcal{R}_y(P_1) \times \mathcal{R}_z(P_2)$ do
        $S \leftarrow C$
        for $I \in \mathcal{I}_{a,b}$ do
            $S \leftarrow S \cap \mathcal{R}_w(I)$
        end for
        $C' \leftarrow C' \cup S$
    end for
    return $C'$
end procedure
```

$\mathcal{R}_w(J_{a,b}) = \bigcap\limits_{I \in \mathcal{I}_{a,b}} \mathcal{R}_w(I)$. 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 4 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.

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

We will now carry out the simulation in Section 4.2 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 $x_i \mapsto \alpha^{x_i}$. $\sum_i a_i x_i$ is thus represented as $\alpha^{\sum_i a_i x_i}$. We will show that all the steps of the simulation in section 4.2 can still be performed after this transformation.

Theorem 5. *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 9. *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 8. *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 4, 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 4, 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 9. *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 $x'_i = x^c_i$. 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 $A^c_1 = A^c$, 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

$$x^2x_2\cdots x_k - a_1xx_2\cdots x_k = 0$$

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

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

Proceeding in a similar way, we can derive

$$x^cx_2\cdots x_k - a_1^cx_2\cdots x_k = 0$$

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 $\sum_i ca_{ij}x_i \geq cb_j$ 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 10. *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

$$(x-1)(x-\alpha)\cdots(x-\alpha^{\sum_i b_{ij}^+}) = 0 \tag{30}$$

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 $2s^2 < p^m - 1$,

$$c \leq s < (p^m - 1)/2s \leq (p^m - 1)/2c \tag{31}$$

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 (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

$$(x - \alpha^{a_1})Q(x) = 0 \tag{32}$$

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 (30), $Q(\alpha^b) \neq 0$. Therefore, using the Substitution lemma with equations (30) and (32) we derive $x - \alpha^{a_1} = 0$ if this term appears in (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

$$(y^c - \alpha^{b_j}) \cdots (y^c - \alpha^{c\sum_i a_{ij}^+}) = 0 \tag{33}$$

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 9 on the above equation, we get

$$(y^c - 1)(y^c - \alpha^c) \cdots (y^c - \alpha^{c\sum_i a_{ij}^+}) = 0 \tag{34}$$

in $\sum_i a_{ij}^+$ lines. Using the Intersection Lemma on equations (33) and (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 5

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

5.1 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 [16] to show that the same can be carried out in Depth-d-PC over $\mathbb{F}_{p^m}$. Below we describe the proof system of [16] and their simulation of $\mathsf{AC}^0[p]$-Frege.

5.1.1 The Proof System of Maciel and Pitassi

Maciel and Pitassi [16] define a proof system with mod p, negation, AND, OR and threshold connectives, based on the system PTK by Buss and Clote [6] 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 $\Gamma \rightarrow \Delta$, 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 [16]

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

5.1.2 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 $\mathbb{F}_{p^m}$.

5.1.3 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 [1], any $\mathsf{AC}^0[p]$ circuit can converted to a depth three circuit of a special form.

Theorem 6. *[1]*
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 [16]. 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 $fl(\varphi_1) \cdots fl(\varphi_k) \rightarrow fl(\varphi'_1) \cdots fl(\varphi'_m)$ 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 [16] states the following

Theorem 7. *[16]*
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 8. *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 11. *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 4, 8 and 9 at each level. □

Lemma 12. *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 11 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 11 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 11. When φ is a flat circuit with a Threshold gate at the top, the following lemma suffices.

Lemma 13. *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

$$(y-1)\cdots(y-\alpha^k) = 0 \tag{35}$$

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

$$Q(y - \alpha^{a_1}) = 0 \tag{36}$$

Multiplying equation (35) with Q if it does not contain the term $(y-\alpha^{a_1})$, else multiplying it with Q_1, we arrive at

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

Using Lemma 2 with equations (35) and (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 14. *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 15. *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 11. 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 16. *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 9) 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

$$(1 - (z_1 - 1)^{p-1})(1 - x_1) \tag{37}$$

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

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

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 (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 17. *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 4)

$$(z_1 - 1)\cdots(z_1 - \alpha^{t+1})(1 - x_1) = (z_2 - 1)\cdots(z_2 - \alpha^{t+1})(1 - x_1) = 0 \tag{38}$$

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 (38), we get the required equation. □

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

5.2 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 7 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 18. *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

$$(1-(y_1-\alpha^r)^{p^m-1})x_1=(1-(y_2-1)^{p^m-1})x_1=0 \tag{39}$$

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

$$(1-(y_1-\alpha^r)^{p^m-1})(x_1-1)=(1-(y_2-\alpha^r)^{p^m-1})(x_1-1)=0 \tag{40}$$

Adding equations (39) and (40) we get the required equation. □

6 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 11 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 19. *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 20. *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 11

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

Using the contraction rule on the above equation, we get

$$(y-\alpha^t)\cdots(y-\alpha^k)(1-y_1)=0 \tag{41}$$

Multiplying the given equation $(y-1)\cdots(y-\alpha^{t-1})=0$ by $(1-y_1)$ and using the Intersection Lemma with equation (41), we get $1-y_1=0$, which is the required equation. □

6.1 Existence of Feasible Interpolation

Bonet, Pitassi and Raz [3] 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 9. *Depth-d-PC does not have feasible interpolation unless Blum integers can be factored by polynomial sized circuits*

7 Dealing with large coefficients

In this section, we show how our proof system can work with polynomial inequalities that may have large coefficients. We first mention how they will be represented, and then define addition and multiplication operations over these representation. We then show that some basic properties of addition and multiplication can be derived within the system. This will be sufficient to carry out simulations of SOS and CP with large coefficients.

7.1 High Level Idea

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 [9] 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 $m = \mathit{poly}(n)$ bits [17]. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$ be the set of n binary numbers, each of length $m = \mathit{poly}(n)$, 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.

Notions from Earlier Simulations. From the previous sections, it is clear that our system is capable of the following.

1. Express a boolean formula φ using an extension variable y, and derive $y(y-1) = 0$.
2. Derive a statement by branching on the value of a boolean variable y. That is, to consider the case of $y = b$, we multiply each line of the premise by $y - (1-b)$, derive the required statement and then combine these cases using the Intersection Lemma.

Thus, the steps of our simulations in this section are of the above two types. The exact details of the simulation should easily follow.

7.2 Definitions

We first lay down some definitions which will be used throughout the simulation.

Definition 10. *Bit vectors*
We represent an integer using its bit representation by introducing a variable for each of its bits. A bit vector $\mathbf{a} = [a_m \cdots a_1]$ representing an integer a in our system is therefore a vector of variables which equal the bits of a, ordered from the most significant to the least significant bit. Define $\mathbf{a}(i) = a_i$.

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. 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 $-\mathbf{b}$ is represented by the vector $\mathbf{b}_1 \oplus \mathbf{1}$, where $\oplus$ is the usual bitwise addition operation defined below. $\mathbf{0}$ is represented by the all *zeros vector. 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.*

Definition 11. *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* $\mathbf{b}(i) \neq 1$*.*

Definition 12. *Bitwise addition* $\oplus$
We define below the operator 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* $c_1 = 0$*.*

For bits y*,* z*,* w*, let* $H(y, z) = y \wedge z$ *and let* $H(y, z, w) = 1$ *if and only if* $y + z + w \geq 2$*.* H *denotes the carry bit generated by adding together up to three bits. The following identities are easily derived in the proof system*

$$H(y, z, w) = H(y, z \oplus w) \oplus H(z, w) \tag{42}$$

$$H(H(y, z \oplus w), H(z, w)) = 0 \tag{43}$$

If c_i *are as defined above, then*

$$c_{i+1} = H(y_i, z_i, c_i) \tag{44}$$

Definition 13. *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*.*

Definition 14. *Set addition*
Let $m_2 = m/\log(m_0)$*. For a constant* $\mathbf{a}$*, partition the bits of* $\mathbf{a}$ *into* m_2 *blocks of length* $\log(m_0)$*. 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*. Therefore by* $L_j(\mathbf{a})$ *we interchangeably refer to the bits or the integer represented by them. Define* $\mathbf{a}^o$ *to be the number 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*,* $\sum_i L_j(\mathbf{a}_i^o) M_i$ *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*

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

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

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

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, 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.

Definition 15. *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 first 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

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

Definition 16. *Representing a line*
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 $\mathcal{S}(\mathbf{a}_1X_1 \cdots \mathbf{a}_kX_k)$.

7.3 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 21. *Given three bit vectors $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{w}$, Depth-d-PC can prove the following*

$$(\mathbf{y} \oplus \mathbf{z}) \oplus \mathbf{w} = \mathbf{y} \oplus (\mathbf{z} \oplus \mathbf{w})$$

Proof. 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 that the total carry received in both cases is equal, i.e. $d_i^{\mathbf{y},\mathbf{z}} + d_i^{\mathbf{w}} = d_i^{\mathbf{z},\mathbf{w}} + d_i^{\mathbf{y}}$. Note that the plus sign in this expression denotes the usual integer addition. This is clearly true for $i = 1$. Suppose for some $i \geq 1$ we have derived that the first $i-1$ bits of the left and right hand side are equal and the total carrys to the first i positions are equal. The i^{th} bit on the left is equal to $\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i) \oplus d_i^{\mathbf{y},\mathbf{z}} \oplus d_i^{\mathbf{w}}$ and by the induction hypothesis this is equal to $\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i) \oplus d_i^{\mathbf{z},\mathbf{w}} \oplus d_i^{\mathbf{y}}$, which is the i^{th} bit on the right. Now, $d_{i+1}^{\mathbf{y},\mathbf{z}} = H(\mathbf{y}(i), \mathbf{z}(i), d_i^{\mathbf{y},\mathbf{z}})$ and $d_{i+1}^{\mathbf{w}} = H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus d_i^{\mathbf{y},\mathbf{z}}, \mathbf{w}(i), d_i^{\mathbf{w}})$. For bits a, b, c, d, e, it is easy to derive the identity

$$H(a, b, c) + H(a \oplus b \oplus c, d, e) = H(a, b, d) + H(a \oplus b \oplus d, c, e)$$

Using the above identity, $d_{i+1}^{\mathbf{y},\mathbf{z}} + d_{i+1}^{\mathbf{w}} = H(\mathbf{y}(i), \mathbf{z}(i), \mathbf{w}(i)) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{y},\mathbf{z}}, d_i^{\mathbf{w}})$ and since H is a symmetric function this is equal to $H(\mathbf{y}(i), \mathbf{z}(i), \mathbf{w}(i)) + H(\mathbf{y}(i) \oplus \mathbf{z}(i) \oplus \mathbf{w}(i), d_i^{\mathbf{z},\mathbf{w}}, d_i^{\mathbf{y}})$. By the above identity, this is equal to $d_{i+1}^{\mathbf{z},\mathbf{w}} + d_{i+1}^{\mathbf{y}}$. □

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

Lemma 22. *For $i \leq n$, $\mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i^o = \mathcal{S}^o(\mathbf{y}_1 \cdots \mathbf{y}_i)$*

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 23. $\mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_i) = \mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_{i-1}) \oplus \mathbf{y}_i$

Proof.

$$\begin{aligned}\mathcal{S}(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i &= \mathcal{S}^e(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathcal{S}^o(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i^e\oplus\mathbf{y}_i^o\\ &= \mathcal{S}^e(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i^e\oplus\mathcal{S}^o(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i^o\end{aligned}$$

The last equation follows from Lemma 21. $\mathcal{S}^o(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i^o = S^o(\mathbf{y}_1\cdots\mathbf{y}_i)$ and $\mathcal{S}^e(\mathbf{y}_1\cdots\mathbf{y}_{i-1})\oplus\mathbf{y}_i^e = S^e(\mathbf{y}_1\cdots\mathbf{y}_i)$ by the previous lemma and this completes the proof. □

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

Lemma 24.

$$\mathcal{S}(\mathbf{y}_1X_1\cdots\mathbf{y}_tX_t)\oplus\mathcal{S}(\mathbf{z}_1X_1\cdots\mathbf{z}_tX_t) = \mathcal{S}((\mathbf{y}_1\oplus\mathbf{z}_1)X_1\cdots(\mathbf{y}_t\oplus\mathbf{z}_t)X_t)$$

Proof. Assume by induction that we have proved the theorem until $t = i-1$. Then, by Lemma 23,

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

Now by using Lemma 23, the last line is equal to $\mathcal{S}((\mathbf{y}_1\oplus\mathbf{z}_1)X_1\cdots(\mathbf{y}_i\oplus\mathbf{z}_i)X_i)$. □

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

Lemma 25. *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 we have

$$\begin{aligned}\mathcal{R}(P)\oplus\mathcal{R}(Q) &= \mathcal{S}(\mathbf{a}_1X_1\cdots\mathbf{a}_tX_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{S}(\mathbf{b}_1X_1\cdots\mathbf{b}_tX_t)\oplus\mathcal{R}(Q_1)\\ &= \mathcal{S}(\mathbf{a}_1X_1\cdots\mathbf{a}_tX_t)\oplus\mathcal{S}(\mathbf{b}_1X_1\cdots\mathbf{b}_tX_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{R}(Q_1)\\ &= \mathcal{S}((\mathbf{a}_1\oplus\mathbf{b}_1)X_1\cdots(\mathbf{a}_t\oplus\mathbf{b}_t)X_t)\oplus\mathcal{R}(P_1)\oplus\mathcal{R}(Q_1)\\ &= \mathcal{R}(P+Q)\end{aligned}$$

where the last two equalities follow from the previous two lemmas.

□

Lemma 26. *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 $\mathbf{w_1} = \mathbf{y_1}\oplus\mathbf{z_1}$. Note that for every index i , $\mathbf{y}(i)\oplus\mathbf{z}(i) = \mathbf{y_1}(i)\oplus\mathbf{z_1}(i)$. Let i_0 be the least index such that $\mathbf{y}(i_0)\oplus\mathbf{z}(i_0) = 0$. Note that $w(i) = w_1(i)$ for all $i \le i_0$. We first derive that for every $i > i_0$, $\mathbf{w_1}(i) = \mathbf{w}(i)\oplus 1$. At position i_0, a carry is generated exactly in one of the sums $\mathbf{y}\oplus\mathbf{z}$ or $\mathbf{y_1}\oplus\mathbf{z_1}$. This clearly implies that $\mathbf{w_1}(i_0+1) = \mathbf{w}(i_0+1)\oplus 1$. Now assume that up to some $i > i_0$ the previous statement is true. If $\mathbf{y}(i)\oplus\mathbf{z}(i) = 1$, then the carry propagates and the statement continues to hold. Else, a carry is again generated in exactly one of the sums and hence the statement continues to hold.

Now consider the vector $\mathbf{w_1}\oplus\mathbf{1}$. Since $w(i_0) = 0$, $\mathbf{w_1}\oplus\mathbf{1}$ flips all the bits of w_1 up to i_0 and hence is the vector obtained by flipping all the bits of w. Therefore,

$$-(\mathbf{w}) = \mathbf{y_1}\oplus\mathbf{z_1}\oplus\mathbf{1}\oplus\mathbf{1} = (-\mathbf{y})\oplus(-\mathbf{z})$$

□

7.4 Non-negative vectors are closed under addition

In this section we show that non-negative vectors of bounded length are closed under addition.

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

Lemma 27. *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 the bit s_1 denote the sign of $\mathbf{y}$ and s_2 denote the sign of $\mathbf{z}$. 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 derive 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 28. *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,

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

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. ☐

We now show that non-negative vectors according to our definition are closed under addition, as long as the total length does not exceed m.

Lemma 29. *Let* $\mathbf{y}$ *and* $\mathbf{z}$ *be two non-negative vectors of length* $\ell < m - 1$*. Then the vector* $\mathbf{w} = \mathbf{y} \oplus \mathbf{z}$ *is non-negative.*

Proof. Since $\mathbf{y}$ and $\mathbf{z}$ are non-negative, $\mathbf{y}(i) = \mathbf{z}(i) = 0$ for $\ell + 1 \leq i \leq m$. Therefore the carry to the position $\ell + 2$ is zero and thus $\mathbf{w}(\ell + 2) = \cdots = \mathbf{w}(m) = 0$. ☐

The following corollary now follows easily from the previous two lemmas.

Corollary 10. *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* $\mathcal{S}(\mathbf{y}_1 \cdots \mathbf{y}_k)$ *is non-negative.*

Lemma 30. *Let* $\mathbf{y}$ *and* $\mathbf{z}$ *be two non-negative vectors of length* ℓ *such that* $3\ell < m - 1$*. Then* $\mathcal{SS}(\mathbf{y}, \mathbf{z}) \geq 0$

Proof. Since $\mathbf{z}$ is non-negative of length ℓ, $\mathbf{z}(i) = 0$ for $\ell + 1 \leq i \leq m$. Therefore,

$$\mathcal{SS}(\mathbf{y}, \mathbf{z}) = \mathcal{S}(b_0 \mathbf{y} \cdots b_{m-1} 2^{m-1} \mathbf{y}) = \mathcal{S}(b_0 \mathbf{y} \cdots b_\ell 2^\ell \mathbf{y})$$

Since each of the vectors $b_0 \mathbf{y}$, $\cdots$ $b_\ell 2^\ell \mathbf{y}$ is of length at most 2ℓ and there are ℓ of them, by the previous corollary, their set addition is non-negative.

☐

7.5 Properties of multiplication

Here we show that multiplication is distributive and can be treated as repeated addition.

Lemma 31. *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 32. *Let y, z be two bits and let $\mathbf{w}$ be a vector. Then,*

$$y\mathbf{w} \oplus z\mathbf{w} = (y \oplus z)\mathbf{w} \oplus H(y, z)2\mathbf{w}$$

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, the i^{th} bit on both sides is the same, and $e_{i+1} = H(c_i, y\mathbf{w}(i) \oplus z\mathbf{w}(i))$.

It is true for the case of $i = 1$ since $\mathbf{w}_2(1) = 0$ and thus the first bit on both sides is equal to $y \oplus z$. Also $e_2 = 0$ 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 $i > 1$. Note that

$$e_i = H(c_{i-1}, y\mathbf{w}(i-1) \oplus z\mathbf{w}(i-1))$$

and

$$\mathbf{w}_2(i) = H(y\mathbf{w}(i-1), z\mathbf{w}(i-1))$$

Therefore by using Identities (42) and (44)

$$e_1 \oplus \mathbf{w}_2(i) = H(c_{i-1}, y\mathbf{w}(i-1), z\mathbf{w}(i-1)) = c_i \tag{45}$$

And by Identity (43)

$$H(e_1, \mathbf{w}_2(i)) = 0 \tag{46}$$

Thus the i^{th} bit on the right hand side is given by

$$e_i \oplus \mathbf{w}_1(i) \oplus \mathbf{w}_2(i) = y\mathbf{w}(i) \oplus z\mathbf{w}(i) \oplus c_i$$

which is equal to the corresponding bit on the left hand side.

Also, the carry to the $(i+1)^{th}$ bit on the right hand side is equal to $H(e_i, \mathbf{w}_1(i), \mathbf{w}_2(i))$ which by the identity (42) is equal to $H(e_i, \mathbf{w}_2(i)) \oplus H(\mathbf{w}_1(i), e_i \oplus \mathbf{w}_2(i))$, and by (45) and (46) this is equal to $H(y\mathbf{w}(i) \oplus z\mathbf{w}(i), c_i)$

$\square$

Lemma 33. *Let $\mathbf{y} = [y_{k-1} \cdots y_0]$ and $\mathbf{z} = [z_{k-1} \cdots z_0]$ be two bit vectors, let $\mathbf{w} = \mathbf{y} \oplus \mathbf{z}$ and let $d_1 X_1$ be a monomial. Then,*

$$\mathcal{SS}(d_1X_1, \mathbf{w}) = \mathcal{SS}(d_1X_1, \mathbf{y}) \oplus \mathcal{SS}(d_1X_1, \mathbf{z})$$

Proof. Assume that the statement is derived for 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 $k-1$. Let e_i be the carry to the i^{th} position in $\mathbf{y} \oplus \mathbf{z}$. Then,

$$\begin{aligned}
&\mathcal{SS}(d_1X_1, \mathbf{y}) \oplus \mathcal{SS}(d_1X_1, \mathbf{z}) \\
&= \mathcal{SS}(d_1X_1, \mathbf{y}_{k-1}) \oplus y_{k-1}2^{k-1}d_1X_1 \oplus \mathcal{SS}(d_1X_1, \mathbf{z}_{k-1}) \oplus z_{k-1}2^{k-1}d_1X_1 \\
&= \mathcal{SS}(d_1X_1, \mathbf{y}_{k-1} \oplus \mathbf{z}_{k-1}) \oplus (y_{k-1} \oplus z_{k-1})2^{k-1}d_1X_1 \oplus H(y_{k-1}, z_{k-1})2^k d_1X_1 \\
&= \mathcal{SS}(d_1X_1, \mathbf{w}_{k-1}) \oplus e_k 2^{k-1}d_1X_1 \oplus (y_{k-1} \oplus z_{k-1})2^{k-1}d_1X_1 \oplus H(y_{k-1}, z_{k-1})2^k d_1X_1 \\
&=^1 \mathcal{SS}(d_1X_1, \mathbf{w}_{k-1}) \oplus (y_{k-1} \oplus z_{k-1} \oplus e_k)2^{k-1}d_1X_1 \oplus H(y_{k-1}, z_{k-1}, e_k)2^k d_1X_1 \\
&=^2 \mathcal{SS}(d_1X_1, \mathbf{w})
\end{aligned}$$

where $=^1$ follows from Lemma 23 and the definition of $\mathcal{SS}$ and $=^2$ follows from Identities (42) and (43).

$\square$

Lemma 34. *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*

$$\mathcal{R}(a_0X_0Q) = \mathcal{SS}(a_0X_0, \mathbf{z})$$

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 bit vector for Q_j. Assume that we have proved the above statement for Q_j, $j < k$. Then by Lemma 23, $\mathbf{z} = \mathbf{z}_{k-1} \oplus \mathbf{a}'_kX_k$. Therefore by Lemma 33 we have

$$\mathcal{SS}(\mathbf{a}_0X_0, \mathbf{z}) = \mathcal{SS}(\mathbf{a}_0X_0, \mathbf{z}_{k-1}) \oplus \mathcal{SS}(\mathbf{a}_0X_0, \mathbf{a}'_kX_k)$$

Since the bit length of $a_0a'_i$ is at most $m-1$, $\mathcal{SS}(\mathbf{a}_0X_0, \mathbf{a}'_kX_k) = \mathbf{a}_0\mathbf{a}'_kX_0X_k$ and by induction, $\mathcal{SS}(\mathbf{a}_0X_0, \mathbf{z}_{k-1}) = \mathcal{R}(a_0X_0Q_{k-1})$. Therefore,

$$\mathcal{SS}(\mathbf{a}_0X_0, \mathbf{z}) = \mathcal{R}(a_0X_0Q_{k-1}) \oplus \mathbf{a}_0\mathbf{a}'_kX_0X_k$$

which is equal to $\mathcal{R}(a_0X_0Q_k)$ by the Distributivity of $\mathcal{R}$.

□

Lemma 35. *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,*

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

Proof. Let $P = a_1M_1 + \cdots + a_kM_k$, $Q = a'_1M_1 + \cdots + a'_kM_k$ and let P_j be the sum of the first $j < k$ terms of P . Let $\mathbf{y}_i$ denote the bit vector corresponding to 2^iP. Then $\mathcal{SS}(\mathbf{y}, \mathbf{z}) = \mathcal{S}(z_0\mathbf{y}_0 \cdots z_{m-1}\mathbf{y}_{m-1})$. Now, note that

$$\mathbf{y}_i = 2^i\mathbf{y}_0 = 2^i\mathcal{S}(\mathbf{a}_1M_1 \cdots \mathbf{a}_kM_k) = \mathcal{S}(2^i\mathbf{a}_1M_1 \cdots 2^i\mathbf{a}_kM_k)$$

Let $\mathbf{y}^j_i = \mathcal{S}(2^i\mathbf{a}_1M_1 \cdots 2^i\mathbf{a}_jM_j)$ for $j < k$. By Lemma 23, $\mathbf{y}_i = \mathbf{y}^{k-1}_i \oplus 2^i\mathbf{a}_kM_k$. Therefore,

$$\begin{aligned}
\mathcal{S}(z_0\mathbf{y}_0 \cdots z_{m-1}\mathbf{y}_{m-1}) &= \mathcal{S}(z_0(\mathbf{y}^{k-1}_0 \oplus \mathbf{a}_kM_k) \cdots z_{m-1}(\mathbf{y}^{k-1}_{m-1} \oplus 2^{m-1}\mathbf{a}_kM_k)) \\
&= \mathcal{S}(z_0\mathbf{y}^{k-1}_0 \oplus z_0\mathbf{a}_kM_k \cdots z_{m-1}\mathbf{y}^{k-1}_{m-1} \oplus z_{m-1}2^{m-1}\mathbf{a}_kM_k) \\
&=^1 \mathcal{S}(z_0\mathbf{y}^{k-1}_0 \cdots z_{m-1}\mathbf{y}^{k-1}_{m-1}) \oplus \mathcal{S}(z_0\mathbf{a}_kM_k \cdots z_{m-1}2^{m-1}\mathbf{a}_kM_k) \\
&=^2 \mathcal{SS}(\mathbf{y}^{k-1}_0, \mathbf{z}) \oplus \mathcal{SS}(\mathbf{a}_kM_k, \mathbf{z}) \\
&=^3 \mathcal{SS}(\mathbf{y}^{k-1}_0, \mathbf{z}) \oplus \mathcal{R}(\mathbf{a}_kM_kQ)
\end{aligned}$$

where $=^1$ follows from Lemma 23, $=^2$ follows from the definition of $\mathcal{SS}$ and $=^3$ follows by Lemma 34. By induction on k, $\mathcal{SS}(\mathbf{y}^{k-1}_0, \mathbf{z}) = \mathcal{R}(P_{k-1}Q)$. The lemma now follows by Distributivity of $\mathcal{R}$. □

Lemma 36. *Let P be a polynomial represented by a vector $\mathbf{y}$. Then $\mathcal{R}(-P) = -\mathbf{y}$.*

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 we have

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

where the last two equalities follow by Lemma 26 and the induction hypothesis. □

7.6 Simulating CP with large coefficients

The addition rule of CP is easily simulated using Lemma 25 and Lemma 29. Multiplication of a vector A by a constant c is equivalent to $\mathcal{SS}(A, c)$ and thus the rule easily follows from Lemmas 35 and 30. The following lemma shows how to simulate the division rule.

Lemma 37. *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 $\mathcal{R}(P-1) \geq 0$ from $\mathcal{R}(P) \geq 0$*

Proof. $\mathcal{R}(P) = \mathcal{S}(\mathbf{a}_1x_1 \cdots \mathbf{a}_nx_n) \oplus (-\mathbf{a}_0)$. Since $\mathbf{a}_1 \cdots \mathbf{a}_n$ are even and $\mathbf{a}_0$ is odd, the leftmost bit of $\mathcal{R}(P)$ is one. Since -1 is represented by the all ones vector, the sum $\mathcal{R}(P) \oplus (-1)$ generates a carry at the leftmost bit and propagates it till the m^{th} bit. As $\mathcal{R}(P) \geq 0$, the m^{th} bit of the sum is equal to zero. □

The division rule can now be implemented easily by first using the above lemma if a_0 is odd and then dropping the last bit of the resultant vector.

7.7 Simulating Dynamic Sum of Squares

Definition 17. *Dynamic Sum of squares (SOS)*
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 α, $\beta \in \mathbb{R}$*
2. *From $R_k = 0$ derive $x_i R_k = 0$ for some $i \in \{1 \cdots n\}$*
3. *$R^2 \geq 0$ for some polynomial $R \in \mathbb{R}[x_1 \cdots x_n]$*
4. *From $R_j \geq 0$ and $R_k \geq 0$ derive $\alpha R_j + \beta R_k \geq 0$ for $\alpha \geq 0$, $\beta \geq 0 \in \mathbb{R}$*
5. *From $R_j \geq 0$ and $R_k \geq 0$ derive $R_j R_k \geq 0$*

Rules 1 and 4 above are easily simulated by Lemmas 25 and 35. For rules 2,3 and 5, we have the following lemma.

Lemma 38. *For two polynomials P and Q with at most m_0 monomials and coefficients bounded by m_1 in absolute value, given $\mathcal{R}(P) \geq 0$ and $\mathcal{R}(Q) \geq 0$, we can derive $\mathcal{R}(PQ) \geq 0$*

Proof. Let $\mathbf{y}$ and $\mathbf{z}$ be bit vectors representing P and Q. Note that by the choice of m, the length ℓ of $\mathbf{y}$ and $\mathbf{z}$ is less than $m - 1$. By Lemma 35, $\mathcal{R}(PQ) = \mathcal{SS}(\mathbf{y}, \mathbf{z})$. And by Lemma 30, we can derive $\mathcal{SS}(\mathbf{y}, \mathbf{z}) \geq 0$ given $\mathbf{y} \geq 0$ and $\mathbf{z} \geq 0$ □

Corollary 11. *For any polynomial P as above, we can derive $P^2 \geq 0$*

Proof. Let $Q = -P$. Consider cases based on the sign bit of $\mathcal{R}(P)$. Assuming it is zero, we have that $P \geq 0$ and by the above theorem we get $P^2 \geq 0$. Else by Lemma 36 we have that $Q \geq 0$ and by the above theorem we again derive $P^2 \geq 0$.
□

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:

1. Find any techique for proving lower bounds on the sizes of polynomial calculus proofs that doesn't go through degrees. More precisely, prove size lower bounds for PC proofs where we view variables as taking values $1, -1$, and replace the axioms $x^2 - x$ with $x^2 - 1$.
2. Prove lower bounds for the system Trinomial-$\Pi\Sigma$-PC.
3. Our simulations 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?

Acknowledgements

The authors would like to thank Paul Beame, Srikanth Srinivasan and Lijie Chen for helpful discussions.

References

[1] 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.

[2] 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.

[3] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.

[4] 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.

[5] 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.

[6] Samuel R Buss and Peter Clote. Cutting planes, connectivity, and threshold logic. *Archive for Mathematical Logic*, 35(1):33–62, 1996.

[7] 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.

[8] 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. ACM, 1996.

[9] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates VS. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[10] Dima Grigoriev and Edward A Hirsch. Algebraic proof systems over formulas. *Theoretical Computer Science*, 303(1):83–102, 2003.

[11] Dima Grigoriev and Nicolai Vorobjov. Complexity of null-and positivstellensatz proofs. *Annals of Pure and Applied Logic*, 113(1-3):153–160, 2001.

[12] Joshua A Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 110–119. IEEE, 2014.

[13] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6):37:1–37:59, 2018.

[14] Pavel Hrubes and Iddo Tzameret. The proof complexity of polynomial identities. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 41–51, 2009.

[15] 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.

[16] 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.

[17] Alexis Maciel and Denis Thérien. Threshold circuits of small majority-depth. *Inf. Comput.*, 146(1):55–83, 1998.

[18] 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.

[19] Toniann Pitassi. Unsolvable systems of equations and proof complexity. In *Proceedings of the International Congress of Mathemeticians, Volume III, Berlin*, pages 451–460, 1998.

[20] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

[21] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *computational complexity*, 17(3):407–457, 2008.

[22] 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.

[23] Alexander A Razborov. Lower bounds for the polynomial calculus. *computational complexity*, 7(4):291–324, 1998.

[24] 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.