On the Complexity of Modulo-q Arguments and the Chevalley–Warning Theorem

Mika Göös
Stanford University, CA, USA
https://theory.stanford.edu/~mika/
goos@stanford.edu

Pritish Kamath
Toyota Technological Institute at Chicago, IL, USA
https://pritishkamath.github.io
pritish@ttic.edu

Katerina Sotiraki
Massachusetts Institute of Technology, Cambridge, MA, USA
http://www.mit.edu/~katesot/
katesot@mit.edu

Manolis Zampetakis
Massachusetts Institute of Technology, Cambridge, MA, USA
http://www.mit.edu/~mzampet/
mzampet@mit.edu

Abstract

We study the search problem class PPA_q defined as a modulo-q analog of the well-known *polynomial parity argument* class PPA introduced by Papadimitriou (JCSS 1994). Our first result shows that this class can be characterized in terms of PPA_p for prime p.

Our main result is to establish that an *explicit* version of a search problem associated to the Chevalley–Warning theorem is complete for PPA_p for prime p. This problem is *natural* in that it does not explicitly involve circuits as part of the input. It is the first such complete problem for PPA_p when $p \geq 3$.

Finally we discuss connections between Chevalley-Warning theorem and the well-studied *short integer solution* problem and survey the structural properties of PPA_q.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Total NP Search Problems, Modulo-q arguments, Chevalley–Warning Theorem

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.19

Funding *Mika Göös*: Work done while at IAS. Supported by NSF grant CCF-1412958.
Pritish Kamath: Work done while at MIT. Supported in part by NSF Awards CCF-1733808 and IIS-1741137 and MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646.
Katerina Sotiraki: Supported in parts by NSF/BSF grant #1350619, an MIT-IBM grant, and a DARPA Young Faculty Award, MIT Lincoln Laboratories and Analog Devices.
Manolis Zampetakis: Supported by a Google PhD Fellowship.

Acknowledgements We thank Christos Papadimitriou, Robert Robere, Dmitry Sokolov and Noah Stephens-Davidowitz for helpful discussions. We also thank anonymous referees for valuable suggestions.

1 Introduction

The study of *total* NP *search problems* (TFNP) was initiated by Megiddo and Papadimitriou [32] and Papadimitriou [33] to characterize the complexity of search problems that have a solution for every input and where a given solution can be efficiently checked for validity. Meggido and Papadimitriou [32] showed that the notion of NP-hardness is inadequate to capture the complexity of total NP search problems. By now, this theory has flowered into a sprawling jungle of widely-studied syntactic complexity classes (such as PLS [28], PPA/PPAD/PPP [33], CLS [18]) that serve to classify the complexities of many relevant search problems.

The goal of identifying *natural*[1] complete problems for these complexity classes lies in the foundation of this sub-field of complexity theory and not only gives a complete picture of the computational complexity of the corresponding search problems, but also provides a better understanding of the complexity classes. Such natural complete problems have also been an essential middle-step for proving the completeness of other important search problems, the same way that the NP-completeness of Sat is an essential middle step in showing the NP-completeness of many other natural problems. Some known natural complete problems for TFNP subclasses are: the PPAD-completeness of NashEquilibrium [17], the PPA-completeness of ConsensusHalving, NecklaceSplitting and HamSandwich problems [20, 21] and the PPP-completeness of natural problems related to lattice-based cryptography [36]. Finally, the theory of total search problems has found connections beyond its original scope to areas like communication complexity and circuit lower bounds [23], cryptography [9, 29, 16] and the Sum-of-Squares hierarchy [30].

Our main result is to identify the first natural complete problem for the classes PPA_q, a variant of the class PPA. We also illustrate the relevance of these classes through connections with important search problems from combinatorics and cryptography.

Class PPA_q. The class PPA_q was defined, in passing, by Papadimitriou [33, p. 520]. It is a modulo-q analog of the well-studied *polynomial parity argument* class PPA (which corresponds to $q = 2$). The class embodies the following combinatorial principle:

If a bipartite graph has a node of degree not a multiple of q,
then there is another such node.

In more detail, PPA_q consists of all total NP search problems reducible[2] to the problem Bipartite_q defined as follows. An instance of this problem is a balanced bipartite graph $G = (V \cup U, E)$, where $V \cup U = \{0,1\}^n$ together with a designated vertex $v^\star \in V \cup U$. The graph G is implicitly given via a circuit C that computes the neighborhood of every node in G. Let $\deg(v)$ be the degree of the node v in G. A valid solution is a node $v \in \{0,1\}^n$ such that, either

▷ $v = v^\star$ satisfying $\deg(v) \equiv 0 \pmod q$ [*Trivial Solution*] ; or

▷ $v \neq v^\star$ satisfying $\deg(v) \not\equiv 0 \pmod q$.

In Section 2 we provide some other total search problems (Lonely_q, Leaf_q) that are reducible to and from Bipartite_q. Any one of these problems could be used to define PPA_q. In fact, Lonely_q and Leaf_q are natural variants of the standard problems Lonely and Leaf which are used to define the class PPA.

[1] Following the terminology of many TFNP papers, including [24, 20, 21, 36], a natural problem is one that does not have explicitly a circuit or a Turing machine as part of the input.

[2] Here, we consider a *many-one reduction*, which is a polynomial time algorithm with one oracle query to the said problem. In contrast, a *Turing reduction* allows polynomially many oracle queries. See Subsection 1.5 for a comparison.

Our contributions. We illustrate the importance of the complexity classes PPA_q by showing that many important search problems whose computational complexity is not well understood belong to PPA_q (see §1.6 for details). These problems span a wide range of scientific areas, from algebraic topology to cryptography. For some of these problems we conjecture that PPA_q-completeness is the right notion to characterize their computational complexity. The study of PPA_q is also motivated from the connections to other important and well-studied classes like PPAD.

In this paper, we provide a systematic study of the complexity classes PPA_q. Our main result is the identification of the first natural complete problem for PPA_q together with some structural results. Below we give a more precise overview of our results.

§1.1 (Details in Section 3): We characterize PPA_q in terms of PPA_p for prime p.

§1.2 (Details in Section 4): Our main result is that an *explicit*[3] version of the Chevalley-Warning theorem is complete for PPA_p for prime p. This problem is *natural* in that it does not involve circuits as part of the input and is the first known natural complete problem for PPA_p when $p \geq 3$.

§1.3 (Details in Section 5): As a consequence of the PPA_p-completeness of our natural problem, we show that restricting the input circuits in the definition of PPA_p to just constant depth arithmetic formulas doesn't change the power of the class.

§1.4 (Details in Section 6): We show a connection between PPA_q and the Short Integer Solution (SIS) problem from the theory of lattices. This connection implies that SIS with constant modulus q belongs to $\mathsf{PPA}_q \cap \mathsf{PPP}$, but also provides a polynomial time algorithm for solving SIS when the modulus q is constant and has only 2 and 3 as prime factors.

§1.5 (Details in Section 7): We sketch how existing results already paint a near-complete picture of the relative power of PPA_p relative to other TFNP subclasses (via inclusions and oracle separations). We also show that PPA_q is closed under Turing reductions.

In §1.6, we include a list of open problems that illustrate the broader relevance of PPA_q. We note that a concurrent and independent work by Hollender [25] also establishes the structural properties of PPA_q corresponding to §1.1 and §1.5.

1.1 Characterization via Prime Modulus

We show, in Section 3, that every class PPA_q is built out of the classes PPA_p for p a prime. To formalize this result, we recall the operator "&" defined by Buss and Johnson [13, §6]. For any two syntactic complexity classes M_0, M_1 with complete problems S_0, S_1, the class M_0 & M_1 is defined via its complete problem S_0 & S_1 where, on input $(x, b) \in \{0,1\}^* \times \{0,1\}$, the goal is to find a solution for x interpreted as an instance of problem S_b. Namely, if $b = 0$ then the output has to be a solution of S_0 with input x, and otherwise it has to be a solution of S_1 with input x. Intuitively speaking, M_1 & M_2 combines the powers of both M_1 and M_2. Note that $\mathsf{M}_1 \cup \mathsf{M}_2 \subseteq \mathsf{M}_1$ & M_2. We can now formally express our characterization result (where $p|q$ is the set of primes p dividing q).

▶ **Theorem 1.** $\mathsf{PPA}_q = \&_{p|q}\, \mathsf{PPA}_p$.

[3] Following the terminology in [8], by *explicit* we mean that the system of polynomials, which is the input of the computational problems we define, are given as a sum of monic monomials.

A special case of Theorem 1 is that $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$ for every prime power p^k. Showing the inclusion $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$ is the crux of our proof. This part of the theorem can be viewed as a total search problem analog of the counting class result of Beigel and Gill [7] stating that $\mathsf{Mod}_{p^k}\mathsf{P} = \mathsf{Mod}_p\mathsf{P}$; "an unexpected result", they wrote at the time. Throughout this paper, we use q to denote any integer ≥ 2 and p to denote a prime integer.

1.2 A Natural Complete Problem via Chevalley-Warning Theorem

There have been several works focusing on completeness results for the class PPA (i.e. PPA_2). Initial works showed the PPA-completeness of (non-natural) total search problems corresponding to topological fixed point theorems [24, 1, 19]. Closer to our paper, Belovs et al. [8] show the PPA-completeness of computational analogs of Combinatorial Nullstellensatz and the Chevalley–Warning Theorem, but which explicitly involve a circuit as part of the input. More recently, breakthrough results showed PPA-completeness of problems without a circuit or a Turing Machine in the input such as Consensus-Halving, Necklace-Splitting and Ham-Sandwich [20, 21] resolving an open problem since the definition of PPA in [33].

Our main contribution is to provide a natural complete problem for PPA_p, for every prime p; thereby also yielding a new complete problem for PPA. Our complete problem is an extension of the problem $\textsc{Chevalley}_p$, defined by Papadimitriou [33], which is a search problem associated to the celebrated Chevalley-Warning Theorem. We first present an abstract way to understand the proof of the Chevalley-Warning Theorem that motivates the definition of our natural complete problem for PPA_p.

1.2.1 Max-Degree Monic Monomials and Proof of Chevalley-Warning Theorem

In 1935, Claude Chevalley [15] resolved a hypothesis stated by Emil Artin, that all finite fields are quasi-algebraically closed. Later, Ewald Warning [37] proved a slight generalization of Chevalley's theorem. This generalized statement is usually referred to as the Chevalley-Warning Theorem (CWT, for short). Despite its initial algebraic motivation, CWT has found profound applications in combinatorics and number theory as we discuss in §1.4 (and Section 6).

We now explain the statement of the Chevalley-Warning Theorem, starting with some notations. For any field $\mathbb{F}$ and any polynomial f in a polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ we use $\deg(f)$ to represent the degree of f. We use $\boldsymbol{x}$ to succinctly denote the set of all variables $(x_1, \ldots, x_n)$ (the number of variables will always be n) and $\boldsymbol{f}$ to succinctly denote a system of polynomials $\boldsymbol{f} = (f_1, \ldots, f_m) \in \mathbb{F}[\boldsymbol{x}]^m$. We will often abuse notations to use $\boldsymbol{x}$ to also denote assignments over $\mathbb{F}_p^n$. For instance, let $\mathcal{V}_{\boldsymbol{f}} \coloneqq \left\{\boldsymbol{x} \in \mathbb{F}_p^n : f_i(\boldsymbol{x}) = 0 \text{ for all } i \in [m]\right\}$ be the set of all common roots of $\boldsymbol{f}$.

▶ **Chevalley-Warning Theorem** ([15, 37]). *For any prime[4] p and polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ satisfying*

$$\sum_{i=1}^{m} \deg(f_i) < n, \qquad \text{(CW Condition)}$$

it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$.

[4] While most of the results in this section generalize to prime powers, we only consider prime fields for simplicity.

Given a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, the key idea in the proof of the Chevalley-Warning Theorem is the polynomial

$$\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \coloneqq \prod_{i=1}^{m} \left(1 - f_i(\boldsymbol{x})^{p-1}\right) \pmod{\{x_i^p - x_i\}_i} .$$

Note that $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. Thus, $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x}\in\mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod{p}$. The following definition informally describes a special type of monomial of $\mathsf{CW}_{\boldsymbol{f}}$ that is of particular interest in the proof. For the precise definition, we refer to Section 4.

▶ **Definition 2** (Max-Degree Monic Monomials (Informal)). *Let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$. A monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ refers to a monic monomial obtained when symbolically expanding $\mathsf{CW}_{\boldsymbol{f}}$ as a sum of monic monomials. A monic monomial is said to be of max-degree if it is $\prod_{j=1}^{n} x_j^{p-1}$.*

In the above definition, it is important to consider the *symbolic expansion* of $\mathsf{CW}_{\boldsymbol{f}}$ and ignore any cancellation of coefficients that might occur. Observe that, although the expansion of $\mathsf{CW}_{\boldsymbol{f}}$ is exponentially large in the description size of $\boldsymbol{f}$, each monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ can be succinctly described as a combination of monic monomials of the polynomials $f_1, \dots, f_m$. We formally discuss this in Section 4.

Using the definition of max-degree monic monomials, we state the main technical lemma underlying the proof of CWT (with proof in Section 4).

▶ **Chevalley–Warning Lemma.** *For any prime p and $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$,*

$$|\mathcal{V}_{\boldsymbol{f}}| \;\equiv\; (-1)^n \cdot |\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\}| \pmod{p} \qquad \text{(CW Lemma)}$$

The Chevalley-Warning Theorem now follows by observing that if $\sum_{i=1}^{m} \deg(f_i) < n$ then the number of max-degree monic monomials of $\mathsf{CW}_{\boldsymbol{f}}$ is zero. Hence, we get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$.

1.2.2 Proofs of Cancellation

From the proof sketch of CWT in the previous section, a slight generalization of CWT follows. In particular, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$ if and only if

$$\left|\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\}\right| \;\equiv\; 0 \pmod{p}, \qquad \text{(Extended CW Condition)}$$

Any condition on $\boldsymbol{f}$ that implies the (Extended CW Condition) can replace (CW Condition) in the Chevalley-Warning Theorem. Note that the (Extended CW Condition) is equivalent to all the max-degree monic monomials in $\mathsf{CW}_{\boldsymbol{f}}$ cancelling out. Thus, we call any such condition on $\boldsymbol{f}$ that implies (Extended CW Condition) to be a "*proof of cancellation*" for the system $\boldsymbol{f}$.

We can now reinterpret the result of Belovs et al. [8] in this framework of "proof of cancellation" conditions. In particular, [8] considers the case $p = 2$ and defines the problem PPA-Circuit-Chevalley, in which a "proof of cancellation" is given in a specific form of circuits. These circuits describe the system $(f_1, \dots, f_m)$ in the PPA-Circuit-Chevalley problem. It is then shown that PPA-Circuit-Chevalley is PPA_2-complete.

1.2.3 Computational Problems Based on Chevalley-Warning Theorem

Every "proof of cancellation" that is *syntactically refutable* can be used to define a total search problem that lies in PPA_p. By *syntactically refutable* we mean that whenever the "proof of cancellation" is false, there exists a small witness that certifies so. In this section, we define three computational problems with their corresponding "proof of cancellation": (1) the $\textsc{Chevalley}_p$ problem defined by [33], (2) the $\textsc{GeneralChevalley}_p$ problem that is a generalization of $\textsc{Chevalley}_p$, and (3) the problem $\textsc{ChevalleyWithSymmetry}_p$ that we show to be PPA_p-complete. All these problems are defined for every prime modulus p and are natural in the sense that they do not explicitly involve a circuit or a Turing Machine in their input. In particular, the polynomial systems in the input are *explicit* in that they are given as a sum of monic monomials.

1.2.3.1 Chevalley

This is the direct computational analog of the Chevalley-Warning Theorem and was defined by Papadimitriou [33] as the following total search problem:

$\underline{\textsc{Chevalley}_p}$
Given an explicit polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$, output one of the following:
▷ [*Refuting witness*] (CW Condition) is not satisfied.
▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \setminus \{\boldsymbol{x}^\star\}$.
We will particularly consider a special case where all the f_i's have zero constant term (*zecote*, for short). In this case, $\boldsymbol{x}^\star = \mathbf{0} \in \mathcal{V}_{\boldsymbol{f}}$, so there is no need to explicitly include $\boldsymbol{x}^*$ in the input.

1.2.3.2 General Chevalley

As mentioned already, we can define a search problem corresponding to any syntactically refutable condition that implies the (Extended CW Condition). One such condition is to directly assert that

$$\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\} \;=\; \emptyset. \qquad \text{(General CW Condition)}$$

In particular, note that (CW Condition) implies this condition. Moreover, this condition is syntactically refutable by a max-degree monic monomial, which is efficiently representable as a combination of at most $m(p-1)$ monomials of the f_i's. Thus, we can define the following total search problem generalizing $\textsc{Chevalley}_p$.

$\underline{\textsc{GeneralChevalley}_p}$
Given an explicit polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$, output one of the following:
▷ [*Refuting Witness*] A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$.
▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \setminus \{\boldsymbol{x}^\star\}$.

While $\textsc{GeneralChevalley}_p$ generalizes $\textsc{Chevalley}_p$, it does not capture the full generality of (Extended CW Condition). However (Extended CW Condition) is not syntactically refutable (in fact, it is $\mathsf{Mod}_p\mathsf{P}$–complete to decide[5] if the final coefficient of the max-degree monomial is 0).

A natural question then is whether $\textsc{GeneralChevalley}_p$, or even $\textsc{Chevalley}_p$, could already be PPA_p–complete. We believe this to be unlikely because (General CW Condition) seems to fail in capturing other simple conditions that are syntactically refutable and yet imply (Extended CW Condition). Namely, consider a permutation permutation $\sigma \in S_n$ of

[5] Circuit-SAT can be encoded as satisfiability of a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ by including a polynomial for each gate along with $\left\{x_i^2 - x_i = 0\right\}$ to ensure Booleanity. Thus, number of satisfiable assignments to the Circuit-SAT is $\equiv |\mathcal{V}_{\boldsymbol{f}}| \pmod p$, which is $0 \pmod p$ iff the final coefficient of the max-degree monomial is 0.

the variables $x_1, \ldots, x_n$ of order p (i.e. σ^p is the identity permutation). Suppose that for every $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{f}}}$, it holds that $\sigma(\boldsymbol{x}) \in \overline{\mathcal{V}_{\boldsymbol{f}}} \smallsetminus \{\boldsymbol{x}\}$; in other words $\boldsymbol{x}, \sigma(\boldsymbol{x}), \sigma^2(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})$ are all distinct and in $\overline{\mathcal{V}_{\boldsymbol{f}}}$ (where, $\sigma(\boldsymbol{x})$ denotes the assignment obtained by permutating the variables of the assignment $\boldsymbol{x}$ according to σ). This implies that the elements of $\overline{\mathcal{V}_{\boldsymbol{f}}}$ can be partitioned into groups of size p (given by the orbits of the action σ) and hence $|\overline{\mathcal{V}_{\boldsymbol{f}}}| \equiv 0 \pmod p$. Hence, such a σ provides a syntactically refutable proof that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$ and hence that (Extended CW Condition) hold.

Hence, we further generalize GeneralChevalley$_p$ into a problem that incorporates this additional "proof of cancellation" in the form of a permutation $\sigma \in S_n$.

1.2.3.3 Chevalley with Symmetry

We consider a union of two polynomial systems $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$. Even if both $\boldsymbol{g}$ and $\boldsymbol{h}$ satisfy (CW Condition), the combined system $\boldsymbol{f} := (g_1, \ldots, g_{m_g}, h_1, \ldots, h_{m_h})$ might not satisfy (CW Condition) and it might even be the case that $|\mathcal{V}_{\boldsymbol{f}}|$ is not a multiple of p. Thus, we need to bring in some additional conditions.

We start by observing that since $|\mathcal{V}_{\boldsymbol{f}}| + |\overline{\mathcal{V}_{\boldsymbol{f}}}| = p^n$, it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$ if and only if $|\overline{\mathcal{V}_{\boldsymbol{f}}}| \equiv 0 \pmod p$. Also note that, $|\overline{\mathcal{V}_{\boldsymbol{f}}}| = |\overline{\mathcal{V}_{\boldsymbol{g}}}| + |(\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}})|$.

If $\boldsymbol{g}$ satisfies the (General CW Condition) then we have that $|\mathcal{V}_{\boldsymbol{g}}| \equiv |\overline{\mathcal{V}_{\boldsymbol{g}}}| \equiv 0 \pmod p$. A simple way to enforce that $|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}| \equiv 0 \pmod p$ is to enforce a "symmetry", namely that its elements can be grouped into groups of size p each. We impose this grouping with a permutation $\sigma \in S_n$ of the variables $x_1, \ldots, x_n$ of order p such that for any $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, it holds that $\sigma(\boldsymbol{x}) \in (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \smallsetminus \{\boldsymbol{x}\}$; or in other words that $\boldsymbol{x}, \sigma(\boldsymbol{x}), \sigma^2(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})$ are all distinct and contained in $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

We now define the following natural total search problem.

ChevalleyWithSymmetry$_p$
Given two explicit polynomial systems $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$, and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$ (where $\boldsymbol{f} := (\boldsymbol{g}, \boldsymbol{h})$) and a permutation $\sigma \in S_n$ of order p, output one of the following:

- ▷ [*Refuting Witness* – 1] A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{g}}$.
- ▷ [*Refuting Witness* – 2] $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \notin (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \smallsetminus \{\boldsymbol{x}\}$.
- ▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \smallsetminus \{\boldsymbol{x}^\star\}$.

The above problem is natural, because the input consists of a system of polynomial in an explicit form, i.e. as a sum of monic monomials, together with a permutation in S_n given say in one-line notation. Also, observe that when $\boldsymbol{h}$ is empty, the above problem coincides with GeneralChevalley$_p$ (since $\overline{\mathcal{V}_{\boldsymbol{h}}} = \emptyset$ when $\boldsymbol{h}$ is empty). Our main result is the following (proved in Section 4).

▶ **Theorem 3.** *For any prime p, ChevalleyWithSymmetry$_p$ is PPA_p-complete.*

1.3 Complete Problems via Small Depth Arithmetic Formulas

While the ChevalleyWithSymmetry$_p$ problem may seem somewhat contrived, the importance of its PPA_p-completeness is illustrated by our next result (proved in Section 5) showing that we can reformulate any of the proposed definitions of PPA_p, by restricting the circuit in the input to be just constant depth arithmetic formulas with gates $\times \pmod p$ and $+ \pmod p$ (we call this class $\mathsf{AC}^0_{\mathbb{F}_p}$). This result is analogous to the NP-completeness of SAT which basically shows that CircuitSAT remains NP-complete even if we restrict the input circuit to be a (CNF) formula of depth 2.

▶ **Theorem 4.** *$\textsc{Lonely}_p$/$\textsc{Bipartite}_p$/$\textsc{Leaf}_p$ with $\mathsf{AC}^0_{\mathbb{F}_p}$ input circuits are PPA_p–complete.*

We hope that this theorem will be helpful in the context of proving PPA_p-hardness of other problems. There it would be enough to consider only constant depth arithmetic formulas (and hence NC^1 Boolean formulas) in the definitions of PPA_p as opposed to unbounded depth circuits. Such a simplification has been a key-step for proving hardness results for other TFNP subclasses, e.g. in the PPAD-hardness proofs of APPROXIMATE-NASH (cf. [35]).

1.4 Applications of Chevalley-Warning

Apart from its initial algebraic motivation, the Chevalley-Warning theorem has been used to derive several non-trivial combinatorial results. Alon et al. [3] show that adding an extra edge to any 4-regular graph forces it to contain a 3-regular subgraph. More generally, they prove that certain types of "almost" regular graphs contain regular subgraphs. Another application of CWT is in proving *zero-sum theorems* similar to the Erdös-Ginzburg-Ziv Theorem. A famous such application is the proof of Kemnitz's conjecture by Reiher [34].

We define two computational problems that we show are reducible to $\textsc{Chevalley}_p$ and suffice for proving most of the combinatorial applications of the Chevalley-Warning Theorem mentioned above (for a certain range of parameters n and m). Both involve finding solutions to a system of linear equations modulo q, given as $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod{q}$ for $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$.

▷ BIS_q: Find $\boldsymbol{x} \in \{0,1\}^n$ satisfying $\boldsymbol{x} \neq \boldsymbol{0}$ and $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod{q}$.

▷ SIS_q: Find $\boldsymbol{x} \in \{-1,0,1\}^n$ satisfying $\boldsymbol{x} \neq \boldsymbol{0}$ and $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod{q}$.

The second problem is a special case of the well-known *short integer solution* problem in ℓ_∞ norm. Note that, when $n > m \cdot \log_2 q$, the totality of SIS_q is guaranteed by pigeonhole principle; that is, SIS_q is in PPP in this range of parameters. We are interested in identifying the range of parameters that places this problem in PPA_q – see Definitions 40 and 41 for the precise range of parameters n and m that we consider. In Theorem 42, we prove a formal version of the following:

▶ **Theorem** (Informal)**.** *For a certain range of parameters n, m, it holds that*

1. *For all primes p : BIS_p and SIS_p are Karp-reducible to $\textsc{Chevalley}_p$, hence are in PPA_p.*
2. *For all q : BIS_q and SIS_q are Turing-reducible to any PPA_q–complete problem.*
3. *For all k : BIS_{2^k} is solvable in polynomial time.*
4. *For k and ℓ : $SIS_{2^k 3^\ell}$ is solvable in polynomial time.*

Even though the SIS_q problem is well-studied in lattice theory, not many results are known in the regime where q is a constant and the number of variables depends linearly on the number of equations. Part (1) of the above theorem establishes a reduction from SIS_p to $\textsc{Chevalley}_p$ for prime p. Part (2) follows by a bootstrapping method that allows us to combine algorithms for SIS_{q_1} and SIS_{q_2} to give an algorithm for $\mathrm{SIS}_{q_1 q_2}$ (for a certain regime for parameters n and m). Finally Parts (3) and (4) results follow by using this bootstrapping method along with the observation that Gaussian elimination provides valid solutions for BIS_2 (hence also SIS_2) and for SIS_3.

1.5 Structural properties

Relation to other classes

Buss and Johnson [13, 27] had defined a class PMOD_q which turns out to be slightly weaker than PPA_q (refer to Section 7). Despite this slight difference between the definitions of PPA_q and PMOD_q, we can still deduce statements about PPA_q from the work of [27]. In particular, it follows that $\mathsf{PPAD} \subseteq \mathsf{PPA}_q$ (refer to Subsection 7.1).

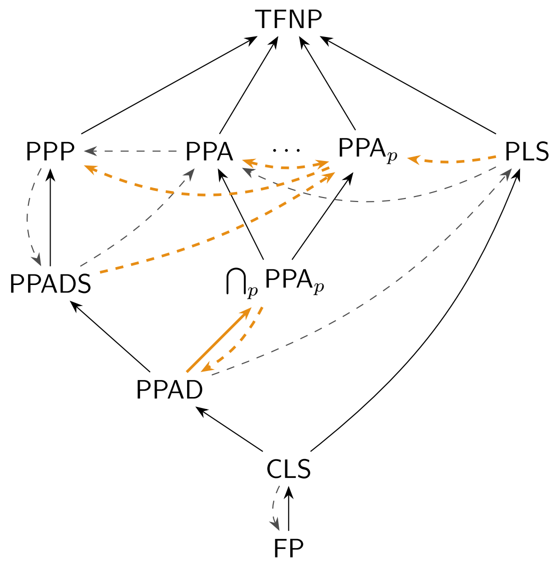


Figure 1 The landscape of TFNP subclasses. A solid arrow $\mathsf{M}_1 \to \mathsf{M}_2$ denotes $\mathsf{M}_1 \subseteq \mathsf{M}_2$, and a dashed arrow $\mathsf{M}_1 \dashrightarrow \mathsf{M}_2$ denotes an oracle separation: $\mathsf{M}_1^{\mathcal{O}} \not\subseteq \mathsf{M}_2^{\mathcal{O}}$ relative to some oracle $\mathcal{O}$. The relationships involving PPA_p are highlighted in yellow. See Section 7 for details.

More broadly, a near-complete picture of the power of PPA_q relative to other subclasses of TFNP is summarized in Figure 1. These relationships (inclusions and oracle separations) mostly follow from prior work in proof complexity [6, 12, 27, 23] (refer to Subsection 7.2).

Closure under Turing reductions

Recall that TFNP subclasses are defined as the set of all total search problems that are *many-one* reducible (aka Karp–reducible) to the corresponding complete problems. One can ask whether more power is gained by allowing *Turing reductions*, that is, polynomially many oracle queries to the corresponding complete problem. Buss and Johnson [13] showed that PLS, PPAD, PPADS, PPA are closed under Turing reductions (with a notable exception of PPP, which remains open). We show this for PPA_p when p is a prime.

▶ **Theorem 5.** $\mathsf{FP}^{\mathsf{PPA}_p} = \mathsf{PPA}_p$ *for every prime* p*.*

By contrast, it follows from [13, §6] that PPA_q is not closed under *black-box* Turing reductions for non-prime powers q. See Subsection 7.3 for details.

1.6 Open questions

Factoring

It has been shown that Factoring reduces to PPP-complete problems as well as to PPA-complete problems [11, 26], albeit under randomized reductions (which can be derandomized assuming the Generalized Reimann Hypothesis). It has been asked whether in fact Factoring could be reduced to PPAD-complete problems [26]. As a step towards this problem, we propose the following question.

▶ **Open Problem 1.** *Is Factoring in* PPA_p *for all primes* p *(perhaps under randomized reductions)?*

This is clearly an easier problem since $\mathsf{PPAD} \subseteq \mathsf{PPA}_p$. Interestingly, note that there exists an oracle $\mathcal{O}$ relative to which $\bigcap_p \mathsf{PPA}_p^{\mathcal{O}} \not\subseteq \mathsf{PPAD}^{\mathcal{O}}$. Thus, the above problem, even if established for all prime p, is still weaker than showing that FACTORING reduces to PPAD-complete problems.

Necklace Splitting

The q-NECKLACE-SPLITTING problem is defined as follows: There is an open necklace[6] with $q \cdot a_i$ beads of color i, for $i \in [n]$. The goal is to cut the necklace in $(q-1) \cdot n$ places and partition the resulting substrings into k collections, each containing precisely a_i beads of color i for each $i \in [n]$.

The fact that such a partition exists was first shown in the case of $q = 2$ by Goldberg and West [22] and by Alon and West [4]. Later, Alon [2] proved it for all $q \geq 2$. As mentioned before, Filos-Ratsikas and Goldberg [21] showed that the 2-NECKLACE-SPLITTING problem is PPA-complete. Moreover, they put forth the following question (which we strengthen further).

▶ **Open Problem 2.** *Is q-NECKLACE-SPLITTING in PPA_q? More strongly, is it PPA_q-complete?*

While we do not know how to prove/disprove this yet, we point out that it was also shown in [21] that 2^k-NECKLACE-SPLITTING is in fact in PPA_2. This is actually well aligned with this conjecture since we showed that $\mathsf{PPA}_{2^k} = \mathsf{PPA}_2$ (Theorem 1).

Bárány-Shlosman-Szücs theorem

Alon's proof of the q-Necklace-Splitting theorem [2] was topological and used a certain generalization of the Borsuk-Ulam theorem due to Bárány, Shlosman and Szücs [14]. Since the computational BORSUK-ULAM problem is PPA-complete, we could ask a similar question about this generalization.

▶ **Open Problem 3.** *Is BÁRÁNY-SHLOSMAN-SZÜCS$_p$ problem in PPA_p (perhaps even PPA_p-complete)?*

Applications of Chevalley-Warning Theorem

We conclude with some interesting directions for further exploring the connections of CHEVALLEY with other computational problems.

▶ **Open Problem 4.** *Does SIS$_q$ admit worst-to-average case reductions to other lattice problems in our range of parameters? Or is it average-case hard assuming standard cryptographic assumptions, e.g. the "learning with errors" assumption?*

If resolved positively, the above would serve as evidence of the average-case hardness for the class PPA_p, similar to the evidence that we have for PPA by reduction from FACTORING.

▶ **Open Problem 5.** *For all primes p, is CHEVALLEY$_p$ reducible to BIS$_p$?*

▶ **Open Problem 6.** *For all q, is there a non-trivial regime of parameters n, m where BIS$_q$ is solvable in polynomial time?*

[6] an "open necklace" means that the beads form a string, not a cycle

2 The class PPA_q

Search Problems in FNP and TFNP

A search problem in FNP is defined by a polynomial time computable relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, that is, for every (x, y), it is possible to decide whether $(x, y) \in \mathcal{R}$ in $\mathrm{poly}(|x|, |y|)$ time. A solution to the search problem on input x is a y such that $|y| = \mathrm{poly}(|x|)$ and $(x, y) \in \mathcal{R}$. For convenience, define $\mathcal{R}(x) \coloneqq \{y : (x, y) \in \mathcal{R}\}$. A search problem is *total* if for every input $x \in \{0,1\}^*$, there exists $y \in \mathcal{R}(x)$ such that $|y| \leq \mathrm{poly}(|x|)$. TFNP is the class of all total search problems in FNP.

Reducibility among search problems

A search problem $\mathcal{R}_1$ is *Karp-reducible* (or *many-one reducible*) to a search problem $\mathcal{R}_2$, or $\mathcal{R}_1 \preceq \mathcal{R}_2$ for short, if there exist polynomial-time computable functions f and g such that given any instance x of $\mathcal{R}_1$, $f(x)$ is an instance of $\mathcal{R}_2$ such that for any $y \in \mathcal{R}_2(f(x))$, it holds that $g(x, f(x), y) \in \mathcal{R}_1(x)$.

On the other hand, we say that $\mathcal{R}_1$ is *Turing-reducible* to $\mathcal{R}_2$, or $\mathcal{R}_1 \preceq_T \mathcal{R}_2$ for short, if there exists a polynomial-time oracle Turing machine that on input x to $\mathcal{R}_1$, makes oracle queries to $\mathcal{R}_2$, and outputs a $y \in \mathcal{R}_1(x)$. In this paper, we primarily deal with Karp-reductions, except in Subsection 7.3, where we compare the two different notions of reductions in the context of PPA_q.

PPA_q via complete problems

We describe several total search problems (parameterized by q) that we show to be inter-reducible. PPA_q is then defined as the set of all search problems reducible to either one of the search problems defined below.

Recall that Boolean circuits take inputs of the form $\{0,1\}^n$ and operate using $(\wedge, \vee, \neg)$ gates. In addition, we'll also consider circuits acting on inputs in $[q]^n$. We interpret the input to be of the form $(\{0,1\}^{\lceil \log q \rceil})^n$, where the circuit will be evaluated only on inputs where each block of $\lceil \log q \rceil$ bits represents a element in $[q]$. In the case where q is a prime, we could also represent the circuit as $C : \mathbb{F}_q^n \to \mathbb{F}_q^n$ with arbitrary gates of the form $g : \mathbb{F}_q^2 \to \mathbb{F}_q$. However, we can simulate any such gate with $\mathrm{poly}(q)$ many $+$ and $\times$ operations (over $\mathbb{F}_q$) along with a constant (1) gate. Hence, in the case of prime q, we'll assume that such circuits are composed of only $(+, \times, 1)$ gates.

▶ **Definition 6** (Bipartite$_q$).

Principle: *A bipartite graph with a non-multiple-of-q degree node has another such node.*

Object: *Bipartite graph* $G = (V \cup U, E)$. *Designated vertex* $v^* \in V$

Inputs: ▹ $C : \{0,1\}^n \to (\{0,1\}^n)^k$, *with* $(\{0,1\}^n)^k$ *interpreted as a k-subset of* $\{0,1\}^n$
▹ $v^* \in \{0\} \times \{0,1\}^{n-1}$ *(usually* 0^n*)*

Encoding: $V \coloneqq \{0\} \times \{0,1\}^{n-1}$, $U \coloneqq \{1\} \times \{0,1\}^{n-1}$,
$E \coloneqq \{(v, u) : v \in V \cap C(u)$ *and* $u \in U \cap C(v)\}$

Solutions: v^* *if* $\deg(v^*) \equiv 0 \pmod q$ *and*
$v \neq v^*$ *if* $\deg(v) \not\equiv 0 \pmod q$

▶ **Definition 7** (Lonely$_q$).

Principle: *A q-dimensional matching on a non-multiple-of-q many vertices has an isolated node.*

Object: *q-dimensional matching $G = (V, E)$.*
Designated vertices $V^* \subseteq V$ with $|V^*| \leq q - 1$

Inputs: $\triangleright$ $C : [q]^n \to [q]^n$
$\triangleright$ $V^* \subseteq [q]^n$ *with* $|V^*| \leq q - 1$

Encoding: $V \coloneqq [q]^n$. *For distinct* $v_1, \ldots, v_q$, *edge* $e \coloneqq \{v_1, \ldots, v_q\} \in E$ *if* $C(v_i) = v_{i+1}$, $C(v_q) = v_1$

Solutions: $v \in V^*$ *if* $\deg(v) = 1$ *and*
$v \notin V^*$ *if* $\deg(v) = 0$

▶ **Definition 8** (Leaf$_q$).

Principle: *A q-uniform hypergraph with a non-multiple-of-q degree node has another such node.*

Object: *q-uniform hypergraph $G = (V, E)$. Designated vertex $v^* \in V$*

Inputs: $\triangleright$ $C : \{0,1\}^n \to (\{0,1\}^{nq})^q$; *Interpret* $(\{0,1\}^{nq})^q$ *as q many q-subsets of* $\{0,1\}^n$
$\triangleright$ $v^* \in \{0,1\}^n$ *(usually 0^n)*

Encoding: $V \coloneqq \{0,1\}^n$. *For distinct* $v_1, \ldots, v_q$, *edge* $e \coloneqq \{v_1, \ldots, v_q\} \in E$ *if* $e \in C(v)$ *for all* $v \in e$

Solutions: v^* *if* $\deg(v) \equiv 0 \pmod{q}$ *and*
$v \neq v^*$ *if* $\deg(v) \not\equiv 0 \pmod{q}$

We remark that Lonely$_q$ and Leaf$_q$ are modulo-q analogs of the PPA-complete problems Lonely and Leaf [33, 5]. We prove the following theorem in Appendix A.

▶ **Theorem 9.** *The problems Bipartite$_q$, Lonely$_q$ and Leaf$_q$ are inter-reducible.*

▶ Remark 10 (Simplifications in describing reductions.). We will use the following simple conventions repeatedly, in order to simplify the descriptions of reductions between different search problems.

1. We will often use “algorithms”, instead of “circuits” to encode our hypergraphs. It is standard to simulate polynomial-time algorithms by polynomial sized circuits.
2. While our definitions require vertex sets to be of a very special form, e.g. $\{0,1\}^n$ or $[q]^n$, it will hugely simplify the description of our reductions to let vertex sets be of arbitrary sizes. This is not a problem as long as the vertex set is efficiently indexable, that is, elements of V must have a poly(n) length representation and we must have a poly-time computable bijective map $\varphi : V \to [|V|]$, whose inverse is also poly-time computable. We could then use φ to interpret the first $|V|$ elements of $\{0,1\}^n$ (or $[q]^n$) as vertices in V. Note that, we need to ensure that no new solutions are introduced in this process. In the case of Bipartite$_q$ or Leaf$_q$, we simply leave the additional vertices isolated and they don't contribute any new solutions. In the case of Lonely$_q$ we need to additionally ensure that $|V| \equiv 0 \pmod{q}$, so that we can easily partition the remaining vertices into q-uniform hyperedges thereby not introducing any new solutions.
3. The above simplification gives us that all our problems have an *instance-extension property* (cf. [10]) – this will be helpful in proving Theorem 5.
4. To simplify our reductions even further, we'll often describe the edges/hyperdges directly instead of specifying how to compute the neighbors of a given vertex. This is only for simplicity and it will be easy to see how to compute the neighbors of any vertex locally.

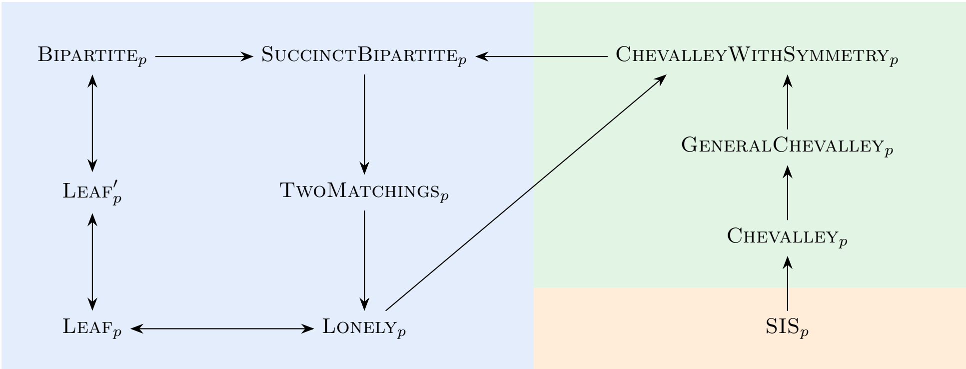


Figure 2 Total search problems studied in this work. An arrow $A \to B$ denotes a reduction $A \preceq B$ that we establish. Problems in the blue region are non-natural problems, which are all complete for PPA_p. Problems in the green region are natural problems of which CHEVALLEYWITHSYMMETRY$_p$ is the one we show to be PPA_p–complete. The problem in the orange region is a cryptographically relevant problem.

3 Characterization via Primes

In this section we prove Theorem 1, namely $\mathsf{PPA}_q = \&_{p|q} \mathsf{PPA}_p$. The theorem follows by combining the following two ingredients.

§3.1: $\mathsf{PPA}_{qr} = \mathsf{PPA}_q \;\&\; \mathsf{PPA}_r$ for any coprime q and r.

§3.2: $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$ for any prime power p^k.

3.1 Coprime case

$\mathsf{PPA}_{qr} \supseteq \mathsf{PPA}_q \;\&\; \mathsf{PPA}_r$

We show that LONELY$_q$ & LONELY$_r$ reduces to LONELY$_{qr}$. Recall that an instance of LONELY$_q$ & LONELY$_r$ is a tuple (C, V^*, b) where (C, V^*) describes an instance of either LONELY$_q$ or LONELY$_r$ as chosen by $b \in \{0,1\}$. Suppose wlog that $b = 0$, so the input encodes a q-dimensional matching $G = (V, E)$ over $V = [q]^n$ with designated vertices $V^* \subseteq V$, $|V^*| \not\equiv 0 \pmod q$. We can construct a qr-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertices $\overline{V} := V \times [r]$ as follows: For every hyperedge $e := \{v_1, \ldots, v_q\} \in E$, we include the hyperedge $e \times [r]$ in $\overline{E}$. We let the designated vertices of $\overline{G}$ be $\overline{V}^* := V^* \times [r]$. Note that $|\overline{V}^*| \not\equiv 0 \pmod{qr}$. It is easy to see that a vertex (v, i) is isolated in G' iff v is isolated in G. This completes the reduction since $\overline{V}$ is efficiently indexable, and the neighbors of any vertex in $\overline{V}$ are locally computable using black-box access to C.

$\mathsf{PPA}_{qr} \subseteq \mathsf{PPA}_q \;\&\; \mathsf{PPA}_r$

We show that BIPARTITE$_{qr}$ reduces to BIPARTITE$_q$ & BIPARTITE$_r$. Our input instance of BIPARTITE$_{qr}$ is a circuit $C : \{0,1\}^n \to (\{0,1\}^n)^k$ that encodes a bipartite graph $G = (V \cup U, E)$ with a designated node $v^* \in V$. If $\deg(v^*) \equiv 0 \pmod{qr}$, then we already have solved the problem and no further reduction is necessary. Otherwise, if $\deg(v^*) \not\equiv 0 \pmod{qr}$, we have, by the coprime-ness of q and r, that either $\deg(v^*) \not\equiv 0 \pmod q$ or $\deg(v^*) \not\equiv 0 \pmod r$. In the first case (the second case is analogous), we can simply view (G, v^*) as an instance of BIPARTITE$_q$, since vertices with degree $\not\equiv 0 \pmod q$ in G are also solutions to BIPARTITE$_{qr}$.

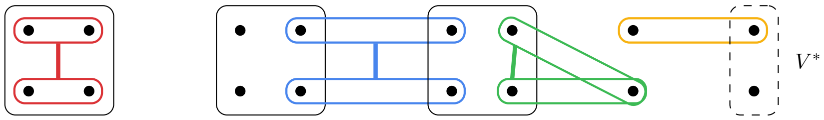


Figure 3 Illustration of the proof of $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$ for $p = 2$, $k = 2$, $n = 2$, $t = 1$. In black, we indicate the 4-dimensional matching G. In color, we highlight some of the vertices of $\overline{G}$ and the edges between them. The vertices of $\overline{G}$ in red, blue and green are paired up and hence are non-solutions; whereas the vertex in yellow is isolated and not in $\overline{V}^*$ and hence a solution.

3.2 Prime power case

$\mathsf{PPA}_{p^k} \supseteq \mathsf{PPA}_p$ follows immediately from our proof of $\mathsf{PPA}_{qr} \supseteq \mathsf{PPA}_q$ & PPA_r, which didn't require that q and r be coprime. It remains to show $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$. We exploit the following easy fact.

▶ **Fact 11.** *For all primes p, it holds that,*

$$\text{for integers } t, c > 0: \quad \binom{c \cdot p^t}{p^t} \equiv 0 \pmod p \quad \text{if and only if} \quad c \equiv 0 \pmod p \tag{3.1}$$

$$\text{for integer } k > 0: \quad \binom{p^k}{i} \equiv 0 \pmod p \quad \text{for all} \quad 0 < i < p^k \tag{3.2}$$

We reduce $\textsc{Lonely}_{p^k}$ to $\textsc{Lonely}_p$. Our instance of $\textsc{Lonely}_{p^k}$ is (C, V^*) where C implicitly encodes a p^k-dimensional matching $G = (V = [p^k]^n, E)$ and a designated vertex set $V^* \subseteq V$ such that $|V^*| \not\equiv 0 \pmod{p^k}$.

Let p^t, $0 \le t < k$, be the largest power of p that divides $|V^*|$. Through local operations we construct a p-dimensional matching hypergraph $\overline{G} = (\overline{V}, \overline{E})$ over vertices $\overline{V} := \binom{V}{p^t}$ (set of all size-p^t subsets of V) with designated vertices $\overline{V}^* := \binom{V^*}{p^t}$. From Eq. 3.1, we get that $|\overline{V}| \equiv 0 \pmod p$ and $|\overline{V}^*| \not\equiv 0 \pmod p$.

We will describe an algorithm that on vertex $\overline{v} \in \overline{V}$ outputs a hyperedge of p vertices that contains $\overline{v}$ (if any). To this end, first fix an algorithm that for any set $e := \{u_1, \ldots, u_{p^k}\} \subseteq V$ and for any $1 \le i \le p^t$, computes some "canonical" partition of the set $\binom{e}{i}$ into subsets of size p, and moreover assigns a canonical cyclic order within each such subset. This is indeed possible because of Eq. 3.2, since $t < k$.

Given a vertex $\overline{v} := \{v_1, \ldots, v_{p^t}\} \in \overline{V}$,
- ▷ Compute all edges $e_1, \ldots, e_\ell \in E$ that include some $v \in \overline{v}$.
- ▷ For edge e_j, define $S_j := e_j \cap \overline{v}$ and let $S_j^1, \ldots, S_j^{p-1}$ be the remaining subsets in the same partition as S_j in the canonical partition of $\binom{e_j}{|S_j|}$, listed in the canonical cyclic order starting at S_j. Also, let S_0 be the set of untouched vertices in $\overline{v}$. Observe that $\overline{v} = S_0 \cup S_1 \cup \ldots \cup S_\ell$.
- ▷ Output neighbors of $\overline{v}$ as the vertices $\overline{v}_1, \ldots, \overline{v}_{p-1}$ where $\overline{v}_i := S_0 \cup S_1^i \cup \ldots \cup S_\ell^i$.

It is easy to see that $\overline{v}$ is isolated in $\overline{G}$ iff all $v \in \overline{v}$ are isolated in G. Moreover, any isolated vertex in $\overline{V} \setminus \overline{V}^*$ contains at least one isolated vertex in $V \setminus V^*$; and a non-isolated vertex in $\overline{V}^*$ contains at least one non-isolated vertex in V^* (in fact p^t many).

The edges of $\overline{G}$ can indeed be computed efficiently with just black-box access to C. In order to complete the reduction, we only need that $\overline{V}$ is efficiently indexable. This is indeed standard; see [31, §2.3] for a reference. See Figure 3 for an illustration of the proof.

▶ Remark 12. Note that the size of the underlying graph blows up polynomially in our reduction. We do not know whether a reduction exists that avoids such a blow-up, although we suspect that the techniques of [6] can be used to show that some blow-up is necessary for black-box reductions.

4 A Natural Complete Problem

We start with some notations that will be useful for the presentation of our results.

Notations. For any polynomial $g \in \mathbb{F}_p[\boldsymbol{x}]$, we define $\deg(g)$ to be the degree of g. We define *the expansion to monic monomials* of g as $\sum_{\ell=1}^{L} t_\ell(\boldsymbol{x})$, where $t_\ell(\boldsymbol{x})$ is a monic monomial in $\mathbb{F}_p[\boldsymbol{x}]$, i.e. a monomial with coefficient 1. For example, the expansion of the polynomial $g(x_1, x_2) = x_1 \cdot (2x_1 + 3x_2)$ is given by $x_1^2 + x_1^2 + x_1x_2 + x_1x_2 + x_1x_2$.

For a polynomial system $\boldsymbol{f} := (f_1, \ldots, f_m) \in \mathbb{F}_p[\boldsymbol{x}]^m$, its affine variety $\mathcal{V}_{\boldsymbol{f}} \subseteq \mathbb{F}_p^n$ is defined as $\mathcal{V}_{\boldsymbol{f}} := \left\{\boldsymbol{x} \in \mathbb{F}_p^n \mid \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}\right\}$. Let $\overline{\mathcal{V}_{\boldsymbol{f}}} := \mathbb{F}_p^n \setminus \mathcal{V}_{\boldsymbol{f}}$. If the constant term of each f_i is 0, we say that $\boldsymbol{f}$ is *zecote*, standing for "Zero Constant Term" (owing to lack of known terminology and creativity on our part).

4.1 The Chevalley-Warning Theorem

We repeat the formal statement of Chevalley-Warning Theorem together with its proof.

▶ **Chevalley-Warning Theorem** ([15, 37]). *For any prime p and a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ satisfying $\sum_{i=1}^m \deg(f_i) < n$ (CW Condition), $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

We describe the proof of CWT through Lemma 14. Even though there are direct proofs, the following presentation helps motivate the generalizations we study in future sections. Given a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, a key idea in the proof is the polynomial $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) := \prod_{i=1}^m \mathsf{CW}_{f_i}(\boldsymbol{x})$ where each $\mathsf{CW}_{f_i}(\boldsymbol{x}) := (1 - f_i(\boldsymbol{x})^{p-1})$. Observe that $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. The following definition describes the notion of a max-degree monomial of $\mathsf{CW}_{\boldsymbol{f}}$ that plays an important role in the proof.

▶ **Definition 13** (Max-Degree Monic Monomials). *For any prime p, let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ and let the expansion into monic monomials of $\mathsf{CW}_{f_i}(\boldsymbol{x})$ be $\sum_{\ell=1}^{r_i} t_{i,\ell}(\boldsymbol{x})$. Let also $U_i = \{(i, \ell) \mid \ell \in [r_i]\}$ and $U = \bigtimes_{i=1}^m U_i$, we define the following quantities.*

1. *A monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ is a product $t_S(\boldsymbol{x}) = \prod_{i=1}^m t_{s_i}(\boldsymbol{x})$ for $S = (s_1, \ldots, s_m) \in U$.*
2. *A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ is any monic monomial $t_S(\boldsymbol{x})$, such that $t_S(\boldsymbol{x}) \equiv \prod_{j=1}^n x_j^{p-1} \left(\bmod\ \{x_i^p - x_i\}_{i\in[n]}\right)$.*
3. *We define $\mathcal{M}_{\boldsymbol{f}}$ to be the set of max-degree monic monomials of $\mathsf{CW}_{\boldsymbol{f}}$, i.e. $\mathcal{M}_{\boldsymbol{f}} := \{S \in U \mid t_S \text{ is a max-degree monic monomial of } \mathsf{CW}_{\boldsymbol{f}}\}$.*

In words, the monomials $t(S)$ are precisely the ones that arise when symbolically expanding $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x})$. We illustrate this with an example: Let $p = 3$ and $f_1(x_1, x_2) = x_1 + x_2$ and $f_2(x_1, x_2) = x_1^2$. Then modulo $\{x_1^3 - x_1, x_2^3 - x_2\}$, we have

$$\begin{aligned}
\mathsf{CW}_{(f_1,f_2)}(x_1, x_2) &= (1 - (x_1 + x_2)^2)(1 - (x_1^2)^2) \\
&= (1 - x_1^2 - 2x_1x_2 - x_2^2) \cdot (1 - x_1^2) \\
&= (1 + x_1^2 + x_1^2 + x_1x_2 + x_2^2 + x_2^2) \cdot (1 + x_1^2 + x_1^2)
\end{aligned}$$

Thus there are 18 $(= 6 \times 3)$ monic monomials in the system (f_1, f_2). The monomial corresponding to $S = ((1, 5), (2, 2))$ is a maximal monomial since the 5-th term in CW_{f_1} is x_2^2 and 2-nd term in CW_{f_2} is x_1^2. Using the above definitions, we now state the main technical lemma of the proof of CWT.

▶ **Lemma 14** (Main Lemma in the proof of CWT). *For any prime p and any system of polynomials $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv (-1)^n |\mathcal{M}_{\boldsymbol{f}}| \pmod p$.*

Proof. As noted earlier, $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. Thus, it follows that $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x} \in \mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod p$. For any monic monomial $m(\boldsymbol{x}) = \prod_{j=1}^n x_j^{d_j}$, it holds that $\sum_{\boldsymbol{x} \in \mathbb{F}_p^n} m(\boldsymbol{x}) = 0$ if $d_j < p-1$ for some x_j. On the other hand, for the monic max-degree monomial $m(\boldsymbol{x}) = \prod_{j=1}^n x_j^{p-1}$, it holds that $\sum_{\boldsymbol{x} \in \mathbb{F}_p^n} m(\boldsymbol{x}) = (p-1)^n$. Thus, we get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x} \in \mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod p \equiv \sum_{S \in U} \sum_{\boldsymbol{x} \in \mathbb{F}_p^n} t_S(\boldsymbol{x}) \pmod p \equiv (-1)^n |\mathcal{M}_{\boldsymbol{f}}| \pmod p$. ◀

The proof of Chevalley-Warning Theorem follows easily from Lemma 14.

Proof of Chevalley-Warning Theorem. We have that $\deg(\mathsf{CW}_{\boldsymbol{f}}) \le (p-1)\sum_{i=1}^m \deg(f_i)$. Thus, if $\boldsymbol{f}$ satisfies (CW Condition), then $\deg(\mathsf{CW}_{\boldsymbol{f}}) < (p-1)n$ and hence $|\mathcal{M}_{\boldsymbol{f}}| = 0$. CWT now follows from Lemma 14. ◀

4.2 The Chevalley-Warning Theorem with Symmetry

In this section, we formalize the intuition that we built in Sections 1.2.2 and 1.2.3 to prove the more general statements to lead to the same conclusion as the Chevalley-Warning Theorem.

First, we prove a theorem that argues about the cardinality of $\mathcal{V}_{\boldsymbol{f}}$ directly using some symmetry of the system of polynomials $\boldsymbol{f}$. Then, combining this symmetry-based argument with the (General CW Condition) we get the generalization of the Chevalley-Warning Theorem. Our natural PPA_p-complete problem is based on this generalization.

The theorem statements are simplified using the definition of *free action* of a group. For a permutation over n elements $\sigma \in S_n$, we define $\langle \sigma \rangle$ to be the sub-group generated by σ and $|\sigma|$ to be the order of $\langle \sigma \rangle$. For $\boldsymbol{x} \in \mathbb{F}_p^n$, $\sigma(\boldsymbol{x})$ denotes the assignment obtained by permutating the variables of the assignment $\boldsymbol{x}$ according to σ.

▶ **Definition 15** (Free Group Action). *Let $\sigma \in S_n$ and $\mathcal{V} \subseteq \mathbb{F}_p^n$, then we say that $\langle \sigma \rangle$ acts freely on $\mathcal{V}$ if, for every $\boldsymbol{x} \in \mathcal{V}$, it holds that $\sigma(\boldsymbol{x}) \in \mathcal{V}$ and $\boldsymbol{x} \neq \sigma(\boldsymbol{x})$.*

Our first theorem highlights the use of symmetry in arguing about the size of $|\mathcal{V}_{\boldsymbol{f}}|$.

▶ **Theorem 16.** *Let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ be a system of polynomials. If there exists a permutation $\sigma \in S_n$ with $|\sigma| = p$ such that $\langle \sigma \rangle$ acts freely on $\overline{\mathcal{V}_{\boldsymbol{f}}}$, then $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

Proof. Since σ acts freely on $\overline{\mathcal{V}_{\boldsymbol{f}}}$, we can partition $\overline{\mathcal{V}_{\boldsymbol{f}}}$ into orbits of any $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{f}}}$ under actions of $\langle \sigma \rangle$, namely sets of the type $\left\{\sigma^i(\boldsymbol{x})\right\}_{i \in [p]}$ for $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{f}}}$. Since $\langle \sigma \rangle$ acts freely on $\overline{\mathcal{V}_{\boldsymbol{f}}}$, each such orbit has size p. Thus, we can conclude that $\left|\overline{\mathcal{V}_{\boldsymbol{f}}}\right| \equiv 0 \pmod p$ from which the theorem follows. ◀

▶ Remark 17. For any polynomial system $\boldsymbol{f}$ and any permutation σ, we can check in linear time if $|\sigma| = p$ and we can syntactically refute that $\langle \sigma \rangle$ acts freely on $\overline{\mathcal{V}_{\boldsymbol{f}}}$ with an $\boldsymbol{x} \in \mathbb{F}_p^n \setminus \{\mathbf{0}\}$ such that $\boldsymbol{f}(\sigma(\boldsymbol{x})) = \mathbf{0}$ or $\sigma(\boldsymbol{x}) = \boldsymbol{x}$.

We now state and prove an extension of CWT that captures both the argument from Lemma 14 and the symmetry argument from Theorem 16.

▶ **Theorem 18** (Chevalley-Warning with Symmetry Theorem). *Let $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$ be two systems of polynomials, and $\boldsymbol{f} := (\boldsymbol{g}, \boldsymbol{h})$. If there exists a permutation $\sigma \in S_n$ with $|\sigma| = p$ such that (1) $\mathcal{M}_{\boldsymbol{g}} = \emptyset$ and (2) $\langle \sigma \rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, then $|\mathcal{V}_{\boldsymbol{f}}| = 0 \pmod p$.*

▶ Remark 19. We point to the special form of Condition 2. By definition, $\mathcal{V}_{\boldsymbol{f}} = \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$, hence if $\langle\sigma\rangle$ were to act freely on $\overline{\mathcal{V}_{\boldsymbol{g}}} \cup \overline{\mathcal{V}_{\boldsymbol{h}}}$ (or even $\mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$), then we could just use Theorem 16 to get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$. In the above theorem, we only require that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. Observe that Theorem 16 follows as a special case of CWT with Symmetry by setting $m_g = 0$. Additionally, by setting $m_h = 0$ we get the generalization of CWT corresponding to the (General CW Condition) as presented in Subsubsection 1.2.3.

Proof of Theorem 18. If $\mathsf{CW}_{\boldsymbol{g}}$ does not have any max- degree monic monomials, we have $|\mathcal{V}_{\boldsymbol{g}}| \equiv 0 \pmod p$ (similar to proof of CWT) and, since $\overline{\mathcal{V}_{\boldsymbol{g}}} = \mathbb{F}_p^n \setminus \mathcal{V}_{\boldsymbol{g}}$, we have $\left|\overline{\mathcal{V}_{\boldsymbol{g}}}\right| \equiv 0 \pmod p$. Also, since $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, we have $\left|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}\right| \equiv 0 \pmod p$ (similar to the proof of Theorem 16). Hence, $\left|\overline{\mathcal{V}_{\boldsymbol{f}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}}} \cup \overline{\mathcal{V}_{\boldsymbol{h}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}}}\right| + \left|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}\right| \equiv 0 \pmod p$. Thus, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$. ◀

4.3 Computational Problems Related to Chevalley-Warning Theorem

We now follow the intuition developed in the previous section and in Subsection 1.2 to formally define the computational problems Chevalley_p, $\text{GeneralChevalley}_p$, and $\text{ChevalleyWithSymmetry}_p$.

▶ **Definition 20** (Chevalley_p)**.**
Principle: *Chevalley-Warning Theorem.*
Input: $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ *: an explicit zecote polynomial system.*
Condition: $\sum_{i=1}^{m} \deg(f_i) < n$.
Output: $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ **Definition 21** ($\text{GeneralChevalley}_p$)**.**
Principle: *General Chevalley-Warning Theorem via* (General CW Condition).
Input: $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ *: an explicit zecote polynomial system.*
Output: ***0.*** *A max-degree monic monomial* $t_S(\boldsymbol{x})$ *of* $\mathsf{CW}_{\boldsymbol{f}}$*, or*
1. $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ **Definition 22** ($\text{ChevalleyWithSymmetry}_p$)**.**
Principle: *Chevalley-Warning Theorem with Symmetry (Theorem 18).*
Input: ▷ $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ *and* $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$ *: explicit zecote polynomial systems*
▷ $\sigma \in S_n$ *: a permutation over* $[n]$.
Condition: $|\sigma| = p$.
Output: ***0. (a)*** *A max-degree monic monomial* $t_S(\boldsymbol{x})$ *of* $\mathsf{CW}_{\boldsymbol{g}}$*, or*
(b) $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ *such that* $\sigma(\boldsymbol{x}) \notin (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \setminus \{\boldsymbol{x}\}$*, or*
1. $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ Remark 23. Some observations about the above computational problems follow:

1. In the problems $\text{GeneralChevalley}_p$ and $\text{ChevalleyWithSymmetry}_p$, we assume that, if the output is a max-degree monic monomial, this is given via the multiset of indices S that describes the monomial as formalized in Definition 13.
2. We have that $\text{Chevalley}_p \preceq \text{GeneralChevalley}_p \preceq \text{ChevalleyWithSymmetry}_p$. Thus, inclusion of $\text{ChevalleyWithSymmetry}_p$ in PPA_p implies that both Chevalley_p and $\text{GeneralChevalley}_p$ are also in PPA_p. Also, in Section 6 we prove that SIS_p reduces to Chevalley_p, where SIS_p is a cryptographically relevant problem. This shows that the problems $\text{GeneralChevalley}_p$ and $\text{ChevalleyWithSymmetry}_p$ are at least as hard as SIS_p.

We restate our main result.

▶ **Theorem 3.** *For any prime p,* $\text{ChevalleyWithSymmetry}_p$ *is* PPA_p*-complete.*

4.4 ChevalleyWithSymmetry$_p$ is PPA$_p$–complete

4.4.1 ChevalleyWithSymmetry$_p$ is in PPA$_p$

Even though Papadimitriou [33] provided a rough proof sketch of Chevalley$_p$ $\in$ PPA$_p$, a formal proof was not given. We show that ChevalleyWithSymmetry$_p$ is in PPA$_p$ (and so are GeneralChevalley$_p$ and Chevalley$_p$). In order to do so we extend the definition of Bipartite$_q$ to instances where the vertices might have exponential degree and edges appear with multiplicity. The key here is to define a Bipartite$_q$ instance with unbounded (even exponential) degree, but with additional information that allows us to verify solutions efficiently.

▶ **Definition 24** (SuccinctBipartite$_q$).

Principle: *Similar to* Bipartite$_q$*, but degrees are allowed to be exponentially large, edges are allowed with multiplicities at most* $q-1$*.*

Object: *Bipartite graph* $G = (V \cup U, E)$ *s.t.* $E \subseteq V \times U \times \mathbb{Z}_q$*. Designated edge* $e^* \in E$*.*

Inputs: *Let* $V \coloneqq \{0\} \times \{0,1\}^{n-1}$ *and* $U \coloneqq \{1\} \times \{0,1\}^{n-1}$*:*

- ▷ $\mathcal{C} : V \times U \to [q]$*, edge counting circuit*
- ▷ $\phi_V : V \times U \times [q] \to (U \times [q])^q$*, grouping pivoted at* V
- ▷ $\phi_U : V \times U \times [q] \to (V \times [q])^q$*, grouping pivoted at* U
- ▷ $e^* = (v^*, u^*, k^*)$*, designated edge*

Encoding: $V \coloneqq \{0\} \times \{0,1\}^{n-1}$, $U \coloneqq \{1\} \times \{0,1\}^{n-1}$,

$E \coloneqq \{(v,u,k) : 1 \le k \le C(v,u),\ (v,u) \in V \times U\}$ *(edges with multiplicities)*

Edge (v,u,k) *is grouped with* $\{(v,u',k') : (u',k') \in \phi_V(v,u,k)\}$ *(pivoting at* v*), provided* $|\phi_V(v,u,k)| = q$*, all* $(v,u',k') \in E$ *and* $\phi_V(v,u',k') = \phi_V(v,u,k)$*.*

Edge (v,u,k) *is grouped with* $\{(v',u,k') : (v',k') \in \phi_U(v,u,k)\}$ *(pivoting at* u*), provided* $|\phi_U(v,u,k)| = q$*, all* $(v,u',k') \in E$ *and* $\phi_U(v',u,k') = \phi_V(v,u,k)$*.*

Solutions: e^* *if* e^* *is grouped, pivoting at* v^**, or if* e^* *is not grouped pivoting at* u^**, OR* $e \ne e^*$ *if* e *is not grouped pivoting at one of its ends.*

In words, SuccinctBipartite$_p$ encodes a bipartite graph with arbitrary degree. Instead of listing the neighbors of a vertex using a circuit, we have a circuit that outputs the multiplicity of edges between any two given vertices. We are therefore unable to efficiently count the number of edges incident on any vertex. The grouping function ϕ_V aims to group edges incident on any vertex $v \in V$ into groups of size q. Similarly, ϕ_U aims to group edges incident on any vertex $u \in U$. The underlying principle is that if we have an edge e^* that is not grouped pivoting at v^* (one of its endpoints), then either e^* is not pivoted at u^* (its other endpoint) or there exists another edge that is also not grouped pivoting at one of its ends. Note that in contrast to the problems previously defined, v^* might still be an endpoint of a valid solution.

▶ **Lemma 25.** *For all primes* p*,* ChevalleyWithSymmetry$_p$ $\in$ PPA$_p$*.*

Proof. We reduce ChevalleyWithSymmetry$_p$ to SuccinctBipartite$_p$, which we show to be PPA$_p$–complete in Subsection A.1. Given an instance of ChevalleyWithSymmetry$_p$, namely a zecote polynomial system $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$ and a permutation σ, we construct a bipartite graph $G = (U \cup V, E)$ encoded as an instance of SuccinctBipartite$_p$ as follows.

Description of vertices. $U = \mathbb{F}_p^n$, namely all possible assignments of $\boldsymbol{x}$. The vertices of V are divided into two parts $V_1 \cup V_2$. The part V_1 contains one vertex for each monomial in the expansion of $\mathsf{CW}_{\boldsymbol{g}} = \prod_{i=1}^{m_g}(1 - g_i^{p-1})$. Since p is constant, we can efficiently list out the monomials of $1 - g_i^{p-1}$. For a fixed lexicographic ordering of the monomials of each $\mathsf{CW}_{g_i} \coloneqq 1 - g_i^{p-1}$, a monomial of $\mathsf{CW}_{\boldsymbol{g}}$ is represented by a tuple $(a_1, a_2, \ldots, a_{m_g})$ with

$0 \le a_i < L_i$, where a_i represents the index of a monomial of CW_{g_i} and L_i is the number of monomials of CW_{g_i}, where $a_i = 0$ corresponds to the constant term 1. The part $V_2 \coloneqq \binom{\mathbb{F}_p^n}{p}$, i.e. it contains a vertex for each subset of p distinct elements in $\mathbb{F}_p^n$.

Description of edges. We first describe the edges between U and V_1, namely include an edge between an assignment $\boldsymbol{x}$ and a monomial t with multiplicity $t(\boldsymbol{x})$. With these edges in place, the degree of vertices are as follows:

- $\boldsymbol{x} = 0^n$ has a single edge corresponding to the constant monomial 1, since $\boldsymbol{f}$ is zecote. We let this be the designated edge e^* in the final $\textsc{SuccinctBipartite}_p$ instance.
- $\boldsymbol{x} \notin \mathcal{V}_{\boldsymbol{g}}$ has $0 \pmod p$ edges (counting multiplicities). Since $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) = 0$, the sum over all monomials of $t(\boldsymbol{x})$ must be $0 \pmod p$.
- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$ has $1 \pmod p$ edges (counting multiplicities), since the sum over all $t(\boldsymbol{x})$ monomials gives $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) \equiv 1 \pmod p$.

Thus with the edges so far, the vertices (excluding 0^n), with degree $\not\equiv 0 \pmod p$ are precisely vertices $t \in V_1$ such that $\sum_{\boldsymbol{x}} t(\boldsymbol{x}) \not\equiv 0 \pmod p$ or $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \setminus \{0^n\}$. For the former case, if t contained a variable with degree less than $p-1$, then $\sum_{\boldsymbol{x}} t(\boldsymbol{x}) \equiv 0 \pmod p$. Hence, it must be that $t = \prod_{i=1}^{n} x_i^{p-1}$. In the later case, the degree of $\boldsymbol{x}$ is $1 \pmod p$ and hence $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$.

However, there is no guarantee that a vertex $\boldsymbol{x}$ with degree $1 \pmod p$ is in $\mathcal{V}_{\boldsymbol{h}}$ as well. To argue about $\boldsymbol{h}$, we add edges between U and V_2 that exclude solutions $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, on which σ acts freely (that is, $\sigma(\boldsymbol{x}) = \boldsymbol{x}$). More specifically, for $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, if $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$, we add an edge with multiplicity $p-1$ between $\boldsymbol{x}$ and $\Sigma_{\boldsymbol{x}} \in V_2$ where $\Sigma_{\boldsymbol{x}} \coloneqq \{\sigma^i(\boldsymbol{x})\}_{i \in \mathbb{Z}_p}$ (note that, in this case $|\Sigma_{\boldsymbol{x}}| = p$ since $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ and $|\sigma| = p$ is prime). Observe that, if a vertex in V_2 corresponds to a $\Sigma_{\boldsymbol{x}}$, it has p edges each with multiplicity $p-1$, one for each $\boldsymbol{x}' \in \Sigma_{\boldsymbol{x}}$ only if $\Sigma_{\boldsymbol{x}} \subseteq \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. If a vertex in V_2 does not correspond to a $\Sigma_{\boldsymbol{x}}$, then it has no edges. Thus, a vertex in V_2 has degree $\not\equiv 0 \pmod p$ iff it contains an $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

Thus, with all the edges added, vertices with degree $\not\equiv 0 \pmod p$ correspond to one of

- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$ such that $\boldsymbol{x} \neq \boldsymbol{0}$, or
- $t \in V_1$ such that $t(\boldsymbol{x})$ is a max-degree monomial or
- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) = \boldsymbol{x}$ or
- $v \in V_2$ such that $\exists\, \boldsymbol{x} \in v$ satisfying $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ and $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

These correspond precisely to the solutions of $\textsc{ChevalleyWithSymmetry}_p$. To summarize, the edge counting circuit C on input $(\boldsymbol{x}, t) \in U \times V_1$ outputs $t(\boldsymbol{x})$ and on input $(\boldsymbol{x}, v) \in U \times V_2$ outputs $p-1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ and $v = \Sigma_{\boldsymbol{x}}$ and 0 otherwise.

Grouping Functions. The grouping functions ϕ_U and ϕ_V are defined as follows (analogous to the so-called "chessplayer algorithm" in [33]):

▷ Grouping ϕ_U (corresponding to endpoint in U):

- For $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{g}}}$: we have that there exists some i such that $\mathsf{CW}_{g_i}(\boldsymbol{x}) = 0$. Consider an edge of the form $(\boldsymbol{x}, (a_1, a_2, \ldots, a_{m_g}), k)$. We can explicitly list out the multiset containing the monomials $t_j = (a_1, a_2, \ldots, a_i \leftarrow j, \ldots, a_{m_g})$ with multiplicity $t_j(\boldsymbol{x})$, for each $1 \le j \le L_i$. Since $\mathsf{CW}_{g_i}(\boldsymbol{x}) = 0$, this multiset has size multiple of p. Hence, we can canonically divide its elements into groups of size p, counting multiplicities and ϕ_U returns the subset containing (t, k).
- For $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$: Note that $g_i^{p-1}(\boldsymbol{x}) = 0$ for all $i \in [m_g]$. Let $v_1 \in V_1$ be the vertex corresponding to the constant monomial 1. ϕ_U groups the edge $(\boldsymbol{x}, v_1, 1)$ (of multiplicity 1) with the $p-1$ edges $(\boldsymbol{x}, \Sigma_{\boldsymbol{x}}, k)$ for $k \in [p-1]$. For any other $t \in V_1 \setminus \{v_1\}$ and an edge $(\boldsymbol{x}, t, k)$, we have that $t = (a_1, \ldots, a_{m_g})$ has $a_i \neq 0$ for some i. We define the multiset containing $t_j = (a_1, \ldots, a_i \leftarrow j, \ldots a_{m_g})$ with multiplicity $t_j(\boldsymbol{x})$ for each $1 \le j < L_i$. Since $g_i^{p-1}(\boldsymbol{x}) = 0$, this multiset has size which is a multiple of p, which we can canonically partition into groups of size p. Thus, ϕ_U on input $(\boldsymbol{x}, t, k)$ returns the group containing (t, k).

▷ Grouping ϕ_V (corresponding to endpoint in V):
- For $t \in V_1$ such that $t \neq \prod_{i=1}^{n} x_i^{p-1}$: there exists a variable x_i with degree less than $p-1$. For $\boldsymbol{x}_j = (x_1, \dots, x_{i-1}, x_i \leftarrow j, \dots, x_n)$ with $j \in \mathbb{F}_p$ we define the multiset $\{(\boldsymbol{x}_j, t(\boldsymbol{x}_j))\}_{j \in \mathbb{F}_p}$. Since $\sum_{j=0}^{p-1} t(\boldsymbol{x}_j) = 0$, this multiset has size multiple of p, so we can canonically partition it into groups of size p. Then, $\phi_V(\boldsymbol{x}, t, k)$ returns the group containing $(\boldsymbol{x}, k)$,
- For $v \in V_2$: if $\deg(v) = 0$, then there is no grouping to be done. Else if $\deg(v) \equiv 0 \pmod p$ then $\phi_V(\boldsymbol{x}, t, k)$ returns $\{(\boldsymbol{x}, k)\}_{\boldsymbol{x} \in v}$.

Thus, for any vertex with degree $\equiv 0 \pmod p$, we have provided a grouping function for all its edges. So, for any edge that is not grouped by grouping function at any of its endpoints, then such an endpoint must have degree $\not\equiv 0 \pmod p$ and hence point to a valid solution of the ChevalleyWithSymmetry$_p$ instance. ◀

4.4.2 ChevalleyWithSymmetry$_p$ is PPA$_p$–hard

We show a reduction from Lonely$_p$ to ChevalleyWithSymmetry$_p$. In the instance of ChevalleyWithSymmetry$_p$ that we create, we will ensure that there are no solutions of type 0 (as in Definition 22) and thus, the only valid solutions will be of type 1. In order to do so, we introduce the notions of labeling and proper labeling and prove a generalization of CWT that we call Labeled CWT (Theorem 30).

As we will see, the Labeled CWT, is just a re-formulation of the original CWT rather than a generalization. To understand the Labeled CWT we start with some examples that do not seem to satisfy the Chevalley-Warning condition, but where a solution exists.

Example 1. Consider the case where $p = 3$ and $f(x_1, x_2) = x_2 - x_1^2$. In this case the Chevalley-Warning condition is not met, since we have 2 variables and the total degree is also 2. But, let us consider a slightly different polynomial where we replace the variable x_2 with the product of two variables x_{21}, x_{22} then we get the polynomial $g(x_1, x_{21}, x_{22}) = x_{21} \cdot x_{22} - x_1^2$. Now, g satisfies (CW Condition) and hence, we conclude that the number of roots of g is a multiple of 3. Interestingly, from this fact we can argue that there exists a non-trivial solution for $f(\boldsymbol{x}) = 0$. In particular, the assignment $x_1 = 0$, $x_2 = 0$ corresponds to five assignments of the variables x_1, x_{21}, x_{22}. Hence, since $|\mathcal{V}_g| = 0 \pmod 3$, g has another root, which corresponds to a non-trivial root of f. In this example, we applied the CWT on a slightly different polynomial than f to argue about the existence of non-trivial solutions of f, even though f did not satisfy (CW Condition) itself.

Ignore Some Terms. The Labeled CWT formalizes the phenomenon observed in Example 1 and shows that under certain conditions we can *ignore some terms* when defining the degree of each polynomial. For instance, in Example 1, we can ignore the term x_1^2 when computing the degree of f and treat f as a degree 1 polynomial of 2 variables, in which case the condition of CWT is satisfied.

We describe which terms can be ignored by defining a *labeling* of the terms of each polynomial in the system. The labels take values in $\{-1, 0, +1\}$ and our final goal is to ignore the terms with label $+1$. Of course, it should not be possible to define any labeling that we want; for example we cannot ignore all the terms of a polynomial. Next, we describe the rules of a *proper labeling* that will allow us to prove the Labeled CWT. We start with a definition of a labeling.

▶ **Definition 26** (Monomial Labeling). *Let $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ and let t_{ij} be the j-th monomial of the polynomial $f_i \in \mathbb{F}[\boldsymbol{x}]$ (written in some canonical sorted order). Let $\mathcal{T}$ be the set of all pairs (i, j) such that t_{ij} is a monomial in $\boldsymbol{f}$. A labeling of $\boldsymbol{f}$ is a function $\lambda : \mathcal{T} \to \{-1, 0, +1\}$ and we say that $\lambda(i, j)$ is the label of t_{ij} according to λ.*

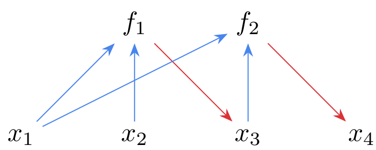
▶ **Definition 27** (Labeled Degree). *For $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ, we define the labeled degree of f_i as, $\deg^\lambda(f_i) := \max_{j\,:\,\lambda(i,j)\neq+1} \deg(t_{ij})$, in words, maximum degree among monomials of f_i labeled either 0 or -1.*

Example 1 (continued). According to the lexicographic ordering, $f(x_1, x_2) = -x_1^2 + x_2$ and we have the monomials $t_{11} = -x_1^2$ and $t_{12} = x_2$. Hence, one possible labeling, which as we will see later corresponds to the vanilla Chevalley-Warning Theorem, is $\lambda(1, 1) = \lambda(1, 2) = 0$. According to this labeling, $\deg^\lambda(f) = 2$. Another possible labeling, that, as we will see, allows us to apply the Labeled CWT , is $\lambda(1, 1) = +1$ and $\lambda(1, 2) = -1$. In this case, the labeled degree is $\deg^\lambda(f) = 1$.

As we highlighted before, our goal is to prove the Chevalley-Warning Theorem, but with the weaker condition that $\sum_{i=1}^m \deg^\lambda(f_i) < n$ instead of $\sum_{i=1}^m \deg(f_i) < n$. Of course, we first have to restrict the space of all possible labelings by defining *proper labelings.* In order to make the condition of proper labelings easier to interpret we start by defining the notion of a labeling graph.

▶ **Definition 28** (Labeling Graph). *For $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ, we define the labeling graph $G_\lambda = (U \cup V, E)$ as a directed bipartite graph on vertices $U = \{x_1, \ldots, x_n\}$ and $V = \{f_1, \ldots, f_m\}$. The edge $(x_j \to f_i)$ belongs to E if x_j appears in a monomial t_{ir} in f_i with label $+1$, i.e. $\lambda(i, r) = +1$. Symmetrically, the edge $(f_i \to x_j)$ belongs to E if the x_j appears in a monomial t_{ir} in f_i with label -1, i.e. $\lambda(i, r) = -1$.*

Example 2. Let $p = 2$ and $f_1(x_1, x_2, x_3, x_4) = x_1x_2 - x_3$, $f_2(x_1, x_2, x_3, x_4) = x_1x_3 - x_4$. In this system, if we use the lexicographic monomial ordering we have the monomials $t_{11} = x_1x_2$, $t_{12} = -x_3$, $t_{21} = x_1x_3$, $t_{22} = -x_4$. The following figure shows the graph G_λ for the labeling $\lambda(1, 1) = +1$, $\lambda(1, 2) = -1$, $\lambda(2, 1) = +1$ and $\lambda(2, 2) = -1$.



▶ **Definition 29** (Proper Labeling). *Let $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ. We say that the labeling λ is proper if the following conditions hold.*

(1) *For all i, either $\lambda(i, j) \in \{-1, 1\}$ for all j, or $\lambda(i, j) = 0$ for all j.*
(2) *If two monomials t_{ij}, $t_{ij'}$ contain the same variable x_k, then $\lambda(i, j) = \lambda(i, j')$.*
(3) *If $\lambda(i, j) = -1$, then t_{ij} is multilinear.*
(4) *If x_k is a variable in the monomials t_{ij}, $t_{i'j'}$, with $i \neq i'$ and $\lambda(i, j) = -1$, then $\lambda(i', j') = +1$.*
(5) *If $\lambda(i, j) \neq 0$, then there exists a j' such that $\lambda(i, j') = -1$.*
(6) *The labeling graph G_λ contains no directed cycles.*

We give an equivalent way to understand the definition of a proper labeling.

- ▷ Condition (1) : there is a partition of the polynomial system $\boldsymbol{f}$ into polynomial systems $\boldsymbol{g}$ and $\boldsymbol{h}$ such that all monomials in $\boldsymbol{g}$ are labeled in $\{+1, -1\}$ and all monomials in $\boldsymbol{h}$ are labeled 0.
- ▷ Condition (2) : each polynomial g_i in $\boldsymbol{g}$ can be written as $g_i = g_i^+ + g_i^-$, such that g_i^+ and g_i^- are polynomials on a disjoint set of variables.
- ▷ Condition (3) : Each g_i^- is multilinear.
- ▷ Condition (4) : Any variable x_k can appear in at most one of the g_i^-. Moreover, if an x_k appears in some g_i^-, it does not appear in any h_j in $\boldsymbol{h}$.
- ▷ Condition (5) : Every g_i^- involves at least one variable.
- ▷ Condition (6) : The graph G_λ is essentially between polynomials in $\boldsymbol{g}$ and the variables that appear in them, with an edge $(g_i \to x_k)$ if x_k appears in g_i^- or an edge $(x_k \to g_i)$ if x_k appears in g_i^+.
- ▷ Note that $\deg^\lambda(g_i) = \deg(g_i^-)$, whereas $\deg^\lambda(h_j) = \deg(h_j)$.

It is easy to see that the trivial labeling $\lambda(i, j) = 0$ is always proper. As we will see this special case of the Labeled CWT corresponds to the original CWT. Note that in this case the labeling graph G_λ is an empty graph. Also, given a system of polynomials $\boldsymbol{f}$ and a labeling λ, it is possible to check in polynomial time whether the labeling λ is proper or not.

Example 2 (continued). It is an instructive exercise to verify that the labeling λ specified was indeed a proper labeling of $\boldsymbol{f}$.

▶ **Theorem 30** (Labeled Chevalley-Warning Theorem). *Let $\mathbb{F}_q$ be a finite field with characteristic p and $\boldsymbol{f} \in \mathbb{F}_q[\boldsymbol{x}]^m$. If λ is a proper labeling of $\boldsymbol{f}$ with $\sum_{i=1}^m \deg^\lambda(f_i) < n$, then $|\mathcal{M}_{\boldsymbol{f}}| = 0$. In particular, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

Proof. Note that $\mathsf{CW}_{\boldsymbol{f}}(x) = \sum_{S \subseteq [m]} \prod_{i \in S} (-1)^{|S|} f_i^{p-1}$. We'll show that every monomial appearing in the expansion of $\prod_{i \in S} f_i^{p-1}$ will have at least one variable with degree at most $p-1$. For simplicity, we focus on the case $S = [m]$ and the other cases of S follow similarly.

We index a monomial of $\prod_{i \in [m]} f_i^{p-1}$ with a tuple

$$((j_{11}, j_{12}, \ldots, j_{1(p-1)}), \ldots, (j_{m1}, \ldots, j_{m(p-1)}))$$

with $1 \le j_{i\ell} \le L_i$ where L_i is the number of monomials in the explicit representation of f_i. The coordinates $(j_{i1}, \ldots, j_{i(p-1)})$ represent the indices of the monomials chosen from each of the $p-1$ copies of f_i^{p-1}. More succinctly, we have $t = \prod_{i=1}^m \prod_{\ell=1}^{p-1} t_{i,j_{i\ell}}$.

Case 1. $\boldsymbol{\lambda(i, j_{i\ell}) \in \{0, -1\}}$, for all $\boldsymbol{(i, \ell)}$:
Here, $\deg(t) \le (p-1) \sum_{i=1}^m \deg^\lambda(f_i)$ which, by our assumption, is strictly less than $(p-1)n$. Hence, there is a variable with degree less than $p-1$.

Case 2. There is a unique $\boldsymbol{i}$ with $\boldsymbol{\lambda(i, j_{i\ell}) = +1}$ for some $\boldsymbol{\ell}$: (warmup for case 3)
That is, for all $i' \neq i$, $\lambda(i', j_{i'\ell}) \in \{0, -1\}$. By condition (5) of proper labeling there exists a $j' \neq j_{i\ell}$ such that $\lambda(i, j') = -1$. Let x_k be a variable in the monomial $t_{ij'}$. By condition (2), x_k is not present in the monomial $t_{i,j_{i\ell}}$ and by condition (3), its degree in $(t_{i,j_{i,1}}, \ldots, t_{i,j_{i,p-1}})$ is at most $p-2$. Additionally, by condition (4), any monomial of $f_{i'}$ for $i' \neq i$ containing x_k must have label $+1$, but $\lambda(i', j_{i',\ell})$ are all in $\{0, -1\}$. Hence, x_k does not appear in any other monomial of t and its degree on t is equal to its degree in $(t_{i,j_{i,1}} \cdots t_{i,j_{i,p-1}})$, which is strictly less than $p-1$.

Case 3. $\boldsymbol{I = \{i : \lambda(i, j_{i\ell}) = +1}$ for some $\boldsymbol{\ell\}}$:
In the labeling graph G_λ, let $i \in I$ be such that there is no path from f_i to any other $f_{i'}$ for $i' \in I$. Such an i exists due to acyclicity of G_λ, i.e. condition (6). Let ℓ be such that $\lambda(i, j_{i\ell}) = +1$. Again, by condition (5) of proper labeling there exists a $j' \neq j_{i\ell}$ such that

$\lambda(i,j') = -1$. Let x_k be a variable in the monomial $t_{ij'}$. By condition (2), x_k is not present in the monomial $t_{i,j_{i\ell}}$ and by condition (3), its degree in $(t_{i,j_{i,1}}, \ldots, t_{i,j_{i,p-1}})$ is at most $p-2$. Additionally, by condition (4), any monomial of $f_{i'}$ for $i' \neq i$ containing x_k must have label $+1$. For $i' \notin I$, $\lambda(i', j_{i',\ell})$ are all in $\{0, -1\}$. And for $i' \in I$, variable x_k cannot appear with $+1$ label in $f_{i'}$ by our choice of f_i. Hence, x_k does not appear in any other monomial of t and its degree on t is equal to its degree on $(t_{i,j_{i,1}} \cdots t_{i,j_{i,p-1}})$, which is strictly less than $p-1$. ◀

We are now ready to prove the PPA_p-hardness of ChevalleyWithSymmetry$_p$.

▶ **Lemma 31.** *For all primes p, ChevalleyWithSymmetry$_p$ is PPA_p-hard.*

Proof. We prove that Lonely$_p \preceq$ ChevalleyWithSymmetry$_p$. Let us assume (without loss of generality from Lemma 44) that the Lonely$_p$ instance has a single distinguished vertex represented by 0^n. We'll assume that 0^n is isolated, otherwise, no further reduction is necessary.

Pre-processing. We slightly modify the given circuit $\mathcal{C}$ by defining $\mathcal{C}' : \mathbb{F}_p^n \to \mathbb{F}_p^n$ as follows:

$$\mathcal{C}'(v) = \begin{cases} v & \text{, if } \mathcal{C}^p(v) \neq v \\ \mathcal{C}(v) & \text{, otherwise} \end{cases}$$

Since p is a prime, a vertex $v \in \mathbb{F}_p^n$ has $\deg(v) = 1$ if and only if $\mathcal{C}^p(v) = v$ and $\mathcal{C}(v) \neq v$. By modifying the circuit, we changed this condition to just $\mathcal{C}'(v) \neq v$, which facilitates our reduction.

Circuit $\mathcal{C}'$ is composed of the $\mathbb{F}_p$-addition $(+)$, $\mathbb{F}_p$-multiplication $(\times)$ and the constant (1) gates. However, we require the input of ChevalleyWithSymmetry$_p$ to be a zecote polynomial system, and so we further modify the circuit $\mathcal{C}'$ to eliminate all the constant (1) gates, without changing it's behavior – this is possible because we assume $\mathcal{C}'(0^n) = 0^n$.

▷ Claim 32. Given circuit $\mathcal{C}'$ with $(+, \times, 1)$ gates, there exists circuit $\bar{\mathcal{C}}$ with $(+, \times)$ gates such that

$$\bar{\mathcal{C}}(\boldsymbol{v}) = \begin{cases} 0^n & \text{, if } \boldsymbol{v} = 0^n \\ \mathcal{C}'(\boldsymbol{v}) & \text{, otherwise} \end{cases}$$

Proof of Claim 32. We replace all instances of the (1) gate by the function $\mathbb{1}_{\{\boldsymbol{v} \neq 0^n\}}$, which we can compute using only $(+, \times)$ gates as follows: For any $x, y \in \mathbb{F}_p$, observe that $\mathbb{1}_{\{x \neq 0\}} \vee \mathbb{1}_{\{y \neq 0\}} = x^{p-1} + y^{p-1} - x^{p-1}y^{p-1}$. We can thus recursively compute $\bigvee_{i=1}^n \mathbb{1}_{\{v_i \neq 0\}}$ using only $(+, \times)$ gates. Thus, $\bar{\mathcal{C}}(\boldsymbol{v}) = \mathcal{C}'(\boldsymbol{v})$ for all $\boldsymbol{v} \neq 0^n$. And $\bar{\mathcal{C}}(0^n) = 0^n$, since $\bar{\mathcal{C}}$ is computed with only $(+, \times)$ gates. ◁

Thus, we can transform our original circuit $\mathcal{C}$ into a circuit $\bar{\mathcal{C}}$ with just $(+, \times)$ gates. For simplicity, we'll write $\bar{\mathcal{C}}$ as simply $\mathcal{C}$ from now on.

As an intermiate step in the reduction we describe a system of polynomials $\boldsymbol{f}_{\mathcal{C}}$ over $2n + s$ variables $(x_1, \ldots, x_n, z_1, \ldots, z_s, y_1, \ldots, y_n)$, where s is the size of the circuit $\mathcal{C}$. The variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ correspond to the input of $\mathcal{C}$, the variables $\boldsymbol{y} = (y_1, \ldots, y_n)$ correspond to the output and the variables $\boldsymbol{z} = (z_1, \ldots, z_s)$ correspond to the gates of $\mathcal{C}$. For an addition gate $(+)$ we include a polynomial of the form

$$f(a_1, a_2, a_3) = a_2 + a_3 - a_1,$$

where a_1 is the variable corresponding to the output of the $(+)$ gate and a_2, a_3 are the variables corresponding to its two inputs. Similarly for a multiplication $(\times)$ gate, we include a polynomial of the form

$$f(a_1, a_2, a_3) = a_2 \cdot a_3 - a_1$$

Finally, for the output of the circuit, we include the polynomial

$$f(a, y_i) = a - y_i,$$

where a is the variable corresponding to the i-th output gate of $\mathcal{C}$. It holds that

$$\mathcal{C}(\boldsymbol{x}) = \boldsymbol{y} \quad \iff \quad \boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \mathbf{0}.$$

We now describe the reduction from $\textsc{Lonely}_p$ to $\textsc{ChevalleyWithSymmetry}_p$. In order to do this, we need to specify a system of polynomials $(\boldsymbol{g}, \boldsymbol{h})$ and a permutation σ such that $|\sigma| = p$. In addition, we will provide a proper labeling λ for $\boldsymbol{g}$ satisfying the degree condition. We will also ensure that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. And hence, the only valid solutions for the resulting $\textsc{ChevalleyWithSymmetry}_p$ instance will be $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$.

Definition of $\boldsymbol{g}$. The polynomial system $\boldsymbol{g}$ contains the following systems of polynomials.

$$\begin{gathered} \boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{1,2}) \\ \boldsymbol{x}_2 - \boldsymbol{x}_3 \\ \boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{z}_{3,4}) \\ \boldsymbol{x}_4 - \boldsymbol{x}_5 \\ \vdots \\ \boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2p-1}, \boldsymbol{x}_{2p}, \boldsymbol{z}_{2p-1,2p}) \end{gathered}$$

Note that there are $N = (2n + s)p$ variables in total.

Labeling $\boldsymbol{\lambda}$ of $\boldsymbol{g}$. For the polynomials belonging to a system of the form $\boldsymbol{f}_{\mathcal{C}}$, the labeling is equal to -1 for the monomials corresponding to the output of each gate and $+1$, otherwise. For instance, let $a_2 + a_3 - a_1$ be the i-th polynomial of $\boldsymbol{g}$ corresponding to a $(+)$ gate and let $a_1 \prec a_2 \prec a_3$, then $\lambda(i, 1) = -1$ and $\lambda(i, 2) = \lambda(i, 3) = +1$.

For the polynomials belonging to a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, the labeling is equal to -1 for the monomials with variables in $\boldsymbol{x}_{i+1}$ and $+1$ for the monomials with variables in $\boldsymbol{x}_i$.

▷ Claim 33. The labeling λ for $\boldsymbol{g}$ is proper.

Proof of Claim 33. By Definition 29, the labeling λ is proper if the following conditions hold.

Condition 1. *For all i, either $\lambda(i, j) \in \{-1, 1\}$ for all j, or $\lambda(i, j) = 0$ for all j.*

In the labeling λ, there are no labels equal to 0, so this condition holds trivially.

Condition 2. *If two monomials t_{ij}, $t_{ij'}$ contain the same variable x_k, then $\lambda(i, j) = \lambda(i, j')$.*

By construction of $\boldsymbol{g}$, no variable appears twice in the same polynomial with a different labeling. For polynomials of $\boldsymbol{f}_{\mathcal{C}}$, this holds because the output variable of a gate is not simultaneously an input variable and all input variables have the same labeling. For polynomials in a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, each polynomial contains two different variables.

Condition 3. *If $\lambda(i,j) = -1$, then t_{ij} is multilinear.*
For polynomials of $\boldsymbol{f}_{\mathcal{C}}$, only the output variable of a gate has label -1 and by definition this monomial is linear. For polynomials in a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, all monomials are linear, so the condition holds trivially.

Condition 4. *If x_k is a variable in the monomials t_{ij}, $t_{i'j'}$, with $i \neq i'$ and $\lambda(i,j) = -1$, then $\lambda(i',j') = +1$.*
Observe that all monomials with label -1 contain only a single variable, so we refer to a monomial x_k with label -1. For a polynomial in $\boldsymbol{f}_{\mathcal{C}}$, a monomial x_k with label -1 corresponds to the output of a gate. Hence, if x_k appears in other monomials of $\boldsymbol{f}_{\mathcal{C}}$, these monomials correspond to inputs and have label $+1$. Also, if x_k is an output variable of $\boldsymbol{f}_{\mathcal{C}}$, then it might appear in a polynomial of the form $a_1 - a_2$. However, by construction the monomials of $x_i - x_{i+1}$ that correspond to output variables of $\boldsymbol{f}_{\mathcal{C}}$ have label $+1$.

Condition 5. *If $\lambda(i,j) \neq 0$, then there exists a j' such that $\lambda(i,j') = -1$.*
By the definition of λ, all polynomials of $\boldsymbol{g}$ have a monomial with label -1. These are the monomials that correspond to the outputs of a gate for the systems of the form $\boldsymbol{f}_{\mathcal{C}}$ and the monomials that correspond to $\boldsymbol{x}_{i+1}$ for the systems of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$.

Condition 6. *The labeling graph G_λ contains no cycles.*
Each system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$ has incoming edges with variables appearing only in the i-th copy of $\boldsymbol{f}_{\mathcal{C}}$ and outgoing edges with variables appearing only in the $(i+1)$-th copy of $\boldsymbol{f}_{\mathcal{C}}$. Also, the variables appearing on the i-th copy of $\boldsymbol{f}_{\mathcal{C}}$ might appear only in the systems $\boldsymbol{x}_{i-1} - \boldsymbol{x}_i$ and $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$. Hence, G_λ has no cycles that contain vertices of two different copies of $\boldsymbol{f}_{\mathcal{C}}$ or of a copy of $\boldsymbol{f}_{\mathcal{C}}$ and a system of the form $\boldsymbol{x}_{i-1} - \boldsymbol{x}_i$.
It is left to argue that the labeling graph restricted to a copy of $\boldsymbol{f}_{\mathcal{C}}$ does not have any cycles. Let the vertices of $\boldsymbol{f}_{\mathcal{C}}$ be ordered according to the topological ordering of $\mathcal{C}$. This restricted part of G_λ corresponds exactly to the graph of $\mathcal{C}$, which by definition is a DAG. Hence, G_λ contains no cycles. ◁

We also need to show that for this labeling $\boldsymbol{g}$ satisfies the labeled Chevalley condition.

▷ Claim 34. The labeled Chevalley condition $\sum_{i=1}^{m_g} \deg^\lambda(g_i) < N$ holds for $\boldsymbol{g}$ with labeling λ.

Proof. Each polynomial of $\boldsymbol{g}$ has a unique monomial with $\lambda(i,j) = -1$ and this monomial has degree 1. Thus, $\sum_{i=1}^{m_g} \deg^\lambda(g_i) = m_g$. On the other hand, the i-th polynomial of $\boldsymbol{g}$ has exactly one variable that has not appeared in any of the previous polynomials. More specifically, the number of variables is equal to $m_g + n$, where n is the size of the input of $\mathcal{C}$. Hence, the labeled Chevalley condition holds for $\boldsymbol{g}$. ◁

Definition of $\boldsymbol{h}$. The system of polynomials $\boldsymbol{g}$ allows us to compute the p vertices given by $\mathcal{C}^i(\boldsymbol{x})$ for $i \in [p+1]$. From the definition of $\textsc{Lonely}_p$ and our pre-processing on $\mathcal{C}$, this group of p vertices is a hyperedge if and only if $\mathcal{C}(\boldsymbol{x}) \neq \boldsymbol{x}$. Since solutions of $\textsc{Lonely}_p$ are lonely vertices, we define $\boldsymbol{h}$ to exclude $\boldsymbol{x}$ such that $\mathcal{C}(\boldsymbol{x}) \neq \boldsymbol{x}$. Namely, we set $\boldsymbol{h}$ to be the system of polynomials

$$\boldsymbol{x}_1 - \boldsymbol{x}_2.$$

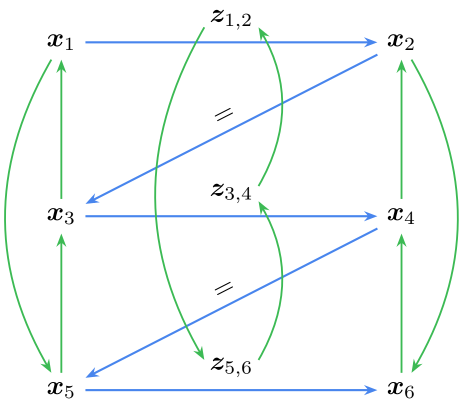
Definition of permutation $\boldsymbol{\sigma}$. In the description of $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$, we have used the following vector of variables:

$$\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{2p}, \boldsymbol{z}_{1,2}, \boldsymbol{z}_{3,4}, \ldots, \boldsymbol{z}_{2p-1,2p})$$

We define the permutation σ such that

$$\sigma(\boldsymbol{x}) = (\boldsymbol{x}_3, \boldsymbol{x}_4, \ldots, \boldsymbol{x}_{2p}, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{3,4}, \boldsymbol{z}_{5,6}, \ldots, \boldsymbol{z}_{2p-1,2p}, \boldsymbol{z}_{1,2}) \ ,$$

as illustrated in the following figure. The blue arrows indicate the polynomials $\boldsymbol{g}$ and the green arrows indicate the permutation σ in the case of $p = 3$.



▷ Claim 35. The group $\langle\sigma\rangle$ has order p and acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

Proof. In order to see that $|\sigma| = p$, note that the input of σ consists of $3p$ blocks of variables. The permutation σ performs a rotation of the first $2p$ blocks by two positions and of the last p blocks by one position.

All that remains is to show that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. First, we show that $\langle\sigma\rangle$ defines a group action on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, that is for all $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, it holds that $\sigma(\boldsymbol{x}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{2p-1}, \boldsymbol{x}_{2p}, \boldsymbol{z}_{1,2}, \boldsymbol{z}_{3,4}, \ldots, \boldsymbol{z}_{2p-1,2p}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, then

- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$ implies that $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2i-1}, \boldsymbol{x}_{2i}, \boldsymbol{z}_{2i-1,2i}) = 0$ for $i \in [p]$ and $\boldsymbol{x}_{2i} = \boldsymbol{x}_{2i+1}$ for $i \in [p-1]$
- $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{h}}}$ implies that $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, that is, $\mathcal{C}(\boldsymbol{x}_1) \neq \boldsymbol{x}_1$ since $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{1,2}) = 0 \Leftrightarrow \boldsymbol{x}_2 = \mathcal{C}(\boldsymbol{x}_1)$.

Now, $\sigma(\boldsymbol{x}) = (\boldsymbol{x}_3, \boldsymbol{x}_4, \ldots, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{3,4}, \boldsymbol{z}_{5,6}, \ldots, \boldsymbol{z}_{1,2}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ holds because

- $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2i-1}, \boldsymbol{x}_{2i}, \boldsymbol{z}_{2i-1,2i}) = 0$ for $i \in [p]$ and $\boldsymbol{x}_{2i} = \boldsymbol{x}_{2i+1}$ for $i \in [p-1]$, which holds from $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$. Additionally, $\boldsymbol{x}_1 = \boldsymbol{x}_{2p}$ holds because we pre-processed $\mathcal{C}$ such that $\mathcal{C}^p(\boldsymbol{x}_1) = \boldsymbol{x}_1$,
- $\boldsymbol{x}_3 \neq \boldsymbol{x}_4$, which holds because $\boldsymbol{x}_4 = \mathcal{C}(\boldsymbol{x}_3)$ for $i \in [p]$ and from the definition of $\mathcal{C}$, $\mathcal{C}(\boldsymbol{x}_1) \neq \boldsymbol{x}_1$ implies that $\boldsymbol{x}_{2i} \neq \boldsymbol{x}_{2i-1}$ for all $i \in [p]$.

Finally, if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, by construction of $\mathcal{C}$, we have that $\boldsymbol{x}_{2k} \neq \boldsymbol{x}_{2j}$ for $k \neq j$ and thus $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ simply because $\boldsymbol{x}_3 \neq \boldsymbol{x}_1$. Thus, we conclude that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. ◁

Putting it all together. The solution of this instance of ChevalleyWithSymmetry$_p$ cannot be a vector $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ with $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ or $\sigma(\boldsymbol{x}) = \boldsymbol{x}$, since we know from Claim 35 that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. We also have from Theorem 30 that the solution also cannot be a max-degree monomial in the expansion of $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) = \prod(1 - g_i^{p-1})$. Thus, the solution must be an $\boldsymbol{x} \neq \boldsymbol{0}$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Let $\boldsymbol{x}_1$ denote the first n coordinates of $\boldsymbol{x}$, then $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ implies that $\boldsymbol{x}_1 = \mathcal{C}(\boldsymbol{x}_1)$ and $\boldsymbol{x} \neq \boldsymbol{0}$ implies that $\boldsymbol{x}_1 \neq \boldsymbol{0}$. Hence, $\boldsymbol{x}_1$ corresponds to a lonely vertex of the Lonely$_p$ instance. ◀

5 Complete Problems via Small Depth Arithmetic Circuits

We now illustrate the significance of the PPA_p-completeness of ChevalleyWithSymmetry$_p$, by showing that we can reformulate any of the proposed definitions of PPA_p, by restricting the circuit in the input to be just constant depth arithmetic formulas with gates $\times \pmod p$ and $+ \pmod p$ (we call this class $\mathsf{AC}^0_{\mathbb{F}_p}$ [7]). This result is analogous to the NP-completeness of SAT which basically shows that CircuitSAT remains NP-complete even if we restrict the input circuit to be a (CNF) formula of depth 2.

We define SuccinctBipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ to be the same as SuccinctBipartite$_p$ but with the input circuit being a formula in $\mathsf{AC}^0_{\mathbb{F}_p}$. Similarly, we define Lonely$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, Leaf$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, etc.

▶ **Theorem 36.** *For all primes p, SuccinctBipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ is PPA_p-complete.*

▶ Remark 37. In [35], a similar simplification theorem was shown for PPAD. In fact, this simplification involves only the End-of-Line problem and does not go through a natural complete problem for PPAD (see Theorem 1.5 in [35]). A similar result can be shown for other TFNP subclasses, including PPA. However, it is unclear if these techniques also apply to PPA_p classes.

Theorem 36 follows directly from the proof of Lemma 25 by observing that the reduction can be perfomed by an $\mathsf{AC}^0_{\mathbb{F}_p}$ circuit. For completeness, we include this proof in Appendix B.

Since the reductions between SuccinctBipartite$_p$ and other problems studied in this work (refer to Appendix A) can also be implemented as AC^0 circuits, we get the following corollary.

▶ **Corollary 38.** *For all primes p, Lonely$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, Leaf$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ and Bipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ are all PPA_p-complete.*

Since $+ \pmod p$ and $\times \pmod p$ can be simulated in NC^1, we also get the following corollary.

▶ **Corollary 39.** *For all primes p, Lonely$_p[\mathsf{NC}^1]$, Leaf$_p[\mathsf{NC}^1]$ and Bipartite$_p[\mathsf{NC}^1]$ are all PPA_p-complete.*

Thus, Theorem 36 allows us to consider reductions from these PPA_p-complete problems with instances encoded by a shallow formulas rather than an arbitrary circuit. We believe this could be a useful starting point for finding other PPA_p-complete problems.

6 Applications of Chevalley-Warning

For most of the combinatorial applications mentioned in Subsection 1.4, the proofs utilize restricted versions of the Chevalley-Warning Theorem that are related to finding binary or short solutions in a system of modular equations. We define two computational problems to capture these restricted cases. The first problem is about finding binary non-trivial solutions in a modular linear system of equations, which we call BIS_q. The second is a special case of the well-known short integer solution problem in ℓ_∞ norm, which we denote by SIS_q. The computational problems are defined below, where $N(q)$ denotes the sum of the exponents in the canonical prime factorization of q, e.g. $N(4) = N(6) = 2$. In particular, $N(p) = 1$ for prime p and $N(q_1 q_2) = N(q_1) + N(q_2)$ for all q_1, q_2.

---

[7] Note that $\mathsf{AC}^0_{\mathbb{F}_p}$ is strictly more powerful than AC^0 since the Boolean operations of $\{\wedge, \vee, \neg\}$ can be implemented in $\mathsf{AC}^0_{\mathbb{F}_p}$, but $+ \pmod p$ cannot be implemented in AC^0.

▶ **Definition 40** (BIS_q).
Input: $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$*, a matrix over* $\mathbb{Z}$
Condition: $n \geq (m+1)^{N(q)}(q-1)$
Output: $\boldsymbol{x} \in \{0,1\}^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod{q}$

▶ **Definition 41** (SIS_q).
Input: $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$*, a matrix over* $\mathbb{Z}$
Condition: $n \geq ((m+1)/2)^{N(q)}(q-1)$
Output: $\boldsymbol{x} \in \{-1,0,1\}^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod{q}$

SIS_q is a special case of the well-known short integer solution problem in ℓ_∞ norm from the theory of lattices. The totality of this problem is guaranteed even when $n > m \log_2 q$ by pigeonhole principle; thus, SIS_q belongs also to PPP (for this regime of parameters). However, for the parameters considered in above definitions, the existence of a solution in the BIS_q and SIS_q is guaranteed through modulo q arguments, which we formally show in the following theorem.

▶ **Theorem 42.** *For the regime of parameters n, m as in Definitions 40 and 41,*
1. *For all primes p : BIS_p, $SIS_p \preceq$ Chevalley$_p$.*
2. *For all q : BIS_q, $SIS_q \in \mathsf{FP}^{\mathsf{PPA}_q}$,*
3. *For all k : $BIS_{2^k} \in \mathsf{FP}$,*
4. *For all k, ℓ : $SIS_{2^k 3^\ell} \in \mathsf{FP}$.*

Proof. Part 1. For all primes p, BIS_p, $\mathrm{SIS}_p \preceq \mathrm{Chevalley}_p$.
Given an BIS_p instance $\boldsymbol{A} = (a_{ij})$, we define a zecote polynomial system as follows

$$\boldsymbol{f} := \left\{ f_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij} x_j^{p-1} \;:\; i \in [m] \right\}$$

Clearly, $\deg(f_i) = p-1$, so $\sum_{i=1}^m \deg(f_i) = m(p-1)$. Since $n \geq (m+1)(p-1) > m(p-1)$, (CW Condition) is satisfied. Hence the output of $\mathrm{Chevalley}_p$ is a solution $\boldsymbol{x} \neq \boldsymbol{0}$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. This gives us that $\boldsymbol{x}^{p-1} := (x_1^{p-1}, \ldots, x_n^{p-1})$ is binary and satisfies $A\boldsymbol{x} \equiv 0 \pmod{p}$.

The reduction $\mathrm{SIS}_p \preceq \mathrm{Chevalley}_p$ also follows in a similar fashion. Namely, we define $f_i(\boldsymbol{x}) := \sum_{j=1}^m a_{ij} x_j^{(p-1)/2}$. This satisfies the (CW Condition) because $\sum_i \deg(f_i) = m(p-1)/2 < ((m+1)/2)(p-1) \leq n$. This ensures that any $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ satisfies $\boldsymbol{x}^{(p-1)/2} \in \{-1,0,1\}^n$ and $A\boldsymbol{x} \equiv 0 \pmod{p}$.

Part 2. For all q : BIS_q, $\mathrm{SIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$.
We show that $\mathrm{BIS}_{q_1 q_2} \preceq \mathrm{BIS}_{q_1}$ & BIS_{q_2}. Hence if $\mathrm{BIS}_{q_1} \in \mathsf{FP}^{\mathsf{PPA}_{q_1}}$ and $\mathrm{BIS}_{q_2} \in \mathsf{FP}^{\mathsf{PPA}_{q_2}}$, then $\mathrm{BIS}_{q_1 q_2} \in \mathsf{FP}^{\mathsf{PPA}_{q_1 q_2}}$. The proof of Part 2 now follows by induction.

Given a $\mathrm{BIS}_{q_1 q_2}$ instance $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$, we divide $\boldsymbol{A}$ along the columns into $n_1 = (m+1)^{N(q_1)}(q_1 - 1)$ submatrices denoted by $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{n_1}$, each of size at least $m \times n_2$, with $n_2 = \lfloor n/n_1 \rfloor$ (if n/n_1 is not an integer, then we let $\boldsymbol{A}_{n_1}$ has more than n_2 columns). Each $\boldsymbol{A}_i$ is an instance of BIS_{q_2}, since

$$n_2 = \lfloor n/n_1 \rfloor \geq (m+1)^{N(q_2)} \lfloor (q-1)/(q_1-1) \rfloor \geq (m+1)^{N(q_2)}(q_2 - 1).$$

Let $\boldsymbol{y}_i \in \{0,1\}^{n_2}$ be any solution to $\boldsymbol{A}_i \boldsymbol{y}_i \equiv 0 \pmod{q_2}$. We define the matrix $\boldsymbol{B} \in \mathbb{Z}^{m \times n_1}$ where the i-th column is equal to $\boldsymbol{A}_i \boldsymbol{y}_i / q_2$; this has integer entries since $\boldsymbol{A}_i \boldsymbol{y}_i \equiv 0 \pmod{q_2}$. Now, by our choice of n_1, we have that $\boldsymbol{B}$ is an instance of BIS_{q_1}. Let $\boldsymbol{z} = (z_1, \ldots, z_{n_1}) \in \{0,1\}^{n_1}$ be any solution to $\boldsymbol{B}\boldsymbol{z} = 0 \pmod{q_1}$.

Finally, we define $\boldsymbol{x} := (z_1\boldsymbol{y}_1, \dots, z_{n_1}\boldsymbol{y}_{n_1}) \in \{0,1\}^n$. Observe that since $\boldsymbol{y}_i$ and $\boldsymbol{z}$ are binary, $\boldsymbol{x}$ is also binary. Additionally,

$$\boldsymbol{A}\boldsymbol{x} \;=\; \sum_{i=1}^{n_1}(\boldsymbol{A}_i\boldsymbol{y}_i)z_i \;=\; q_2\sum_{i=1}^{n_1}\frac{\boldsymbol{A}_i\boldsymbol{y}_i}{q_2}z_i \;=\; q_2\boldsymbol{B}\boldsymbol{y} \;\equiv\; \boldsymbol{0} \pmod{q_1q_2}.$$

Hence, $\boldsymbol{x}$ is a solution of the original $\mathrm{BIS}_{q_1q_2}$ instance $\boldsymbol{A}\boldsymbol{x} \equiv 0 \pmod{q_1q_2}$. This concludes the proof of $\mathrm{BIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$. The proof of $\mathrm{SIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$ follows similarly, by observing that if $\boldsymbol{y}_i$ and $\boldsymbol{z}$ have entries in $\{-1,0,1\}$ then so does $\boldsymbol{x}$.

Parts 3, 4. For all k, ℓ : $\mathrm{BIS}_{2^k} \in \mathsf{FP}$ and $\mathrm{SIS}_{2^k3^\ell} \in \mathsf{FP}$.

Observe that BIS_2 (hence also SIS_2) and SIS_3 are solvable in polynomial time via Gaussian elimination. Combining this with the reduction $\mathrm{BIS}_{q_1q_2} \preceq \mathrm{BIS}_{q_1}$ & BIS_{q_2} completes the proof (similarly for SIS). ◀

Note that for a prime p and any k, we have from Theorem 1, that $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$. Additionally, Theorem 5 shows that PPA_p is closed under Turing reductions, so we have the following corollary.

▶ **Corollary 43.** *For all primes p and all k : BIS_{p^k}, $SIS_{p^k} \in \mathsf{PPA}_p$.*

Even though the SIS_q problem is well-studied in lattice theory, not many results are known in the regime we consider where q is a constant. Our results show that solving $\textsc{Chevalley}_p$ is at least as hard as finding short integer solutions in p-ary lattices for a specific range of parameters. More specifically, our reduction assumes that q is a constant and, thus, it does not depend on the input lattice, and that the dimension n of lattice is related to the number of constraints in the dual as $n > ((m+1)/2)^{N(q)}(q-1)$. On the other hand, we showed (in Parts 3, 4) that there are q-ary lattice for which finding short integer solutions is easy.

7 Structural Properties of PPA_q

In this section, we prove the structural properties of PPA_q outlined in Subsection 1.5.

Relation to PMOD_q

Buss and Johnson [13, 27] defined a problem $\textsc{Mod}_q$, which is almost identical to $\textsc{Lonely}_q$, with the only difference being that the q-dimensional matching is over a power-of-2 many vertices encoded by $C : \{0,1\}^n \to \{0,1\}^n$, with no designated vertices, except when q is a power of 2 in which case we have one designated vertex. The class PMOD_q is then defined as the class of total search problems reducible to $\textsc{Mod}_q$. The restriction of number of vertices to be a power of 2, which arises as an artifact of the binary encoding of circuit inputs, makes the class PMOD_q slightly weaker than PPA_q.

To compare PPA_q and PMOD_q, we define a restricted version of $\textsc{Lonely}_q$, where the number of designated vertices is exactly k; call this problem $\textsc{Lonely}_q^k$. Clearly, $\textsc{Lonely}_q^k$ reduces to $\textsc{Lonely}_q$. We show that a converse holds, but only for prime p; see Subsection A.2 for proof.

▶ **Lemma 44.** *For all primes p and $k \in \{1, \dots, p-1\}$, $\textsc{Lonely}_p$ reduces to $\textsc{Lonely}_p^k$.*

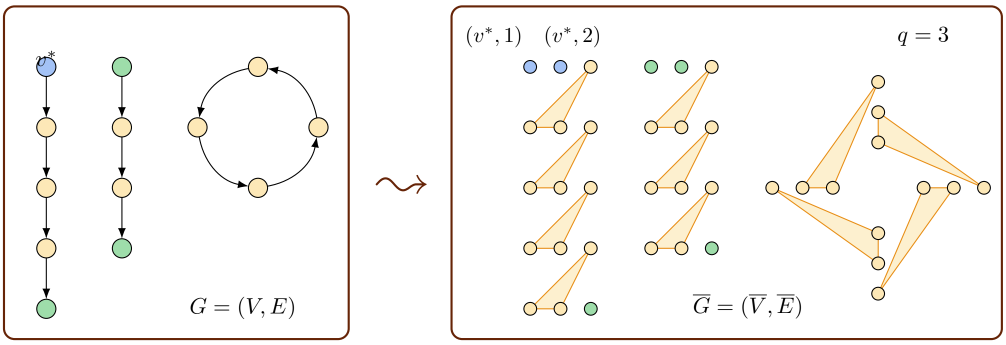
▶ **Corollary 45.** *For all primes p, $\mathsf{PPA}_p = \mathsf{PMOD}_p$.*

For composite q, however, the two classes are conceivably different. In contrast to Theorem 1, it is shown in [27] that $\mathsf{PMOD}_q = \parr_{p|q}\ \mathsf{PMOD}_p$, where the operator "$\parr$" is defined as follows: For any two search problem classes M_0, M_1 with complete problems S_0, S_1, the class $\mathsf{M}_0 \parr \mathsf{M}_1$ is defined via the complete problem $S_0 \parr S_1$ defined as follows: Given $(x_0, x_1) \in \Sigma^* \times \Sigma^*$, find a solution to either x_0 interpreted as an instance of S_0 or to x_1 interpreted as an instance of S_1. In other words, $\mathsf{M}_1 \parr \mathsf{M}_2$ is no more powerful than either M_1 or M_2. In particular, it holds that $\mathsf{M}_1 \parr \mathsf{M}_2 = \mathsf{M}_1 \cap \mathsf{M}_2$, whereas $\mathsf{M}_1\ \&\ \mathsf{M}_2 \supseteq \mathsf{M}_1 \cup \mathsf{M}_2$. Because of this distinction, unlike Theorem 1, the proof of $\mathsf{PMOD}_{p^k} = \mathsf{PMOD}_p$ in [27] follows much more easily since for any odd prime p it holds that $2^n \not\equiv 0 \pmod{p}$ and hence a $\textsc{Lonely}_{p^k}$ instance readily reduces to a $\textsc{Lonely}_p$ instance.

7.1 $\mathsf{PPAD} \subseteq \mathsf{PPA}_q$

Johnson [27] already showed that $\mathsf{PPAD} \subseteq \mathsf{PMOD}_q$ which implies that $\mathsf{PPAD} \subseteq \mathsf{PPA}_q$. We present a simplified version of that proof.

We reduce the PPAD-complete problem $\textsc{End-of-Line}$ to $\textsc{Lonely}_q$. An instance of $\textsc{End-of-Line}$ is a circuit C that implicitly encodes a directed graph $G = (V, E)$, with in-degree and out-degree at most 1 and a designated vertex v^* with in-degree 0 and out-degree 1.



We construct a q-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertices $\overline{V} = V \times [q]$, such that for every edge $(u \to v) \in E$, we include the hyperedge $\{(u, q), (v, 1), \ldots, (v, q-1)\}$ in $\overline{E}$. The designated vertices are $\overline{V}^* = \{(v^*, 1), \ldots, (v^*, q-1)\}$. Note that $|\overline{V}| \equiv 0 \pmod{q}$ and $|\overline{V}^*| = q - 1 \not\equiv 0 \pmod{q}$. It is easy to see that a vertex (v, i) is isolated in $\overline{G}$ if and only if v is a source or a sink in G. This completes the reduction, since $\overline{V}$ is efficiently representable and indexable and the neighbors of any vertex in $\overline{V}$ are locally computable using black-box access to C (see Remark 10).

7.2 Oracle separations

Here we explain how PPA_q can be separated from other TFNP classes relative to oracles, as summarized in Figure 1. That is, for distinct primes p, p', there exist oracles $O_1, \ldots, O_5$ such that

(1) $\mathsf{PLS}^{O_1} \not\subseteq \mathsf{PPA}_p^{O_1}$ (2) $\mathsf{PPA}_p^{O_2} \not\subseteq \mathsf{PPP}^{O_2}$ (3) $\mathsf{PPA}_{p'}^{O_3} \not\subseteq \mathsf{PPA}_p^{O_3}$

(4) $\mathsf{PPADS}^{O_4} \not\subseteq \mathsf{PPA}_p^{O_4}$ (5) $\bigcap_p \mathsf{PPA}_p^{O_5} \not\subseteq \mathsf{PPAD}^{O_5}$

The usual technique for proving such oracle separations is propositional proof complexity (together with standard diagonalization arguments) [5, 10, 13]. The main insight is that if a problem S_1 reduces to another problem S_2 in a black-box manner, then there are "efficient proofs" of the totality of S_1 starting from the totality of S_2. The discussion below assumes some familiarity with these techniques.

$\mathsf{PLS}^{O_1} \not\subseteq \mathsf{PPA}_p^{O_1}$, $\mathsf{PPA}_p^{O_2} \not\subseteq \mathsf{PPP}^{O_2}$, $\mathsf{PPA}_{p'}^{O_3} \not\subseteq \mathsf{PPA}_p^{O_3}$

Johnson [27] showed all the above separations with respect to PMOD_p. Since we showed $\mathsf{PPA}_p = \mathsf{PMOD}_p$ (Corollary 45), the same oracle separations hold for PPA_p.

$\mathsf{PPADS}^{O_4} \not\subseteq \mathsf{PPA}_p^{O_4}$

Göös et al. [23, §4.3] building on [6] showed that the contradiction underlying the PPADS-complete search problem Sink-of-Line requires $\mathbb{F}_p$-Nullstellensatz refutations of high degree. This yields the oracle separation.

$\bigcap_p \mathsf{PPA}_p^{O_5} \not\subseteq \mathsf{PPAD}^{O_5}$

For a fixed $k \geq 1$, consider the problem $S_k \coloneqq \parr_{i\in[k]} \text{Lonely}_{p_i}$ where p_i are the primes. Buss et al. [12] showed that the principle underlying S_i is incomparable with the principle underlying $\text{Lonely}_{p_{i+1}}$. This translates into an relativized separation $\bigcap_{i\in[k]} \mathsf{PPA}_{p_i} \not\subseteq \mathsf{PPA}_{p_{i+1}}$ which in particular implies $\bigcap_{i\in[k]} \mathsf{PPA}_{p_i} \not\subseteq \mathsf{PPAD}$. Finally, one can consider the problem $S \coloneqq S_{k(n)}$ where $k(n)$ is a slowly growing function of the input size n. This problem is in $\bigcap_p \mathsf{PPA}_p$ since for each fixed p and for large enough input size, S reduces to the PPA_p-complete problem. On the other hand, the result of Buss et al. [12] is robust enough to handle a slowly growing $k(n)$; we omit the details.

7.3 Closure under Turing reductions

Theorem 5 says that for any prime p, the class PPA_p is closed under Turing reductions. In contrast, Buss and Johnson showed that PPA_{p_1} & PPA_{p_2}, for distinct primes p_1 and p_2, is not closed under *black-box* Turing reductions [13, 27]. In particular, they define the '$\otimes$' operator as follows. For two total search problems S_1 and S_2, the problem $S_1 \otimes S_2$ is defined as: Given $(x_0, x_1) \in \Sigma^* \times \Sigma^*$, find a solution to both x_0 (instance of S_0) and to x_1 (instance of S_1). Clearly the problem $\text{Lonely}_{p_1} \otimes \text{Lonely}_{p_2}$ can be solved with two queries to the oracle PPA_{p_1} & PPA_{p_2}. However, Buss and Johnson [13, 27] show that $\text{Lonely}_{p_1} \otimes \text{Lonely}_{p_2}$ cannot be solved with one oracle query to PPA_{p_1} & PPA_{p_2} under *black-box* reductions. In particular, this implies that PPA_q is not closed under *black-box* Turing reductions, when q is not a prime power. We now prove Theorem 5, which is equivalent to the following.

▶ **Theorem 46.** *For any prime p and total search problem S, if $S \preceq_T \text{Lonely}_p$, then $S \preceq_m \text{Lonely}_p$.*

Proof. The key reason why this theorem holds for prime p is Lemma 44: In a Lonely_p instance, we can assume w.l.o.g. that there are exactly $p-1$ distinguished vertices.

On instance x of the problem S, suppose the oracle algorithm sequentially makes at most $t = \text{poly}(|x|)$ queries to Lonely_p oracle. The i-th query consists of a tuple (C_i, V_i^*) where C_i encodes a p-dimensional matching graph $G_i = (V_i, E_i)$ and $V_i^* \subseteq V_i$ is the set of $p-1$ designated vertices, and let $y_i \in V_i$ be the solution returned by the Lonely_p oracle. The query (C_i, V_i^*) is computable in polynomial time, given x and valid solutions to all previous queries. Finally, after receiving all answers the algorithm returns $L(x, y_1, \ldots, y_t)$ that is a valid solution for x in S.

We make the following simplifying assumptions.

- Each hypergraph G_i is on p^n vertices, where $n = \text{poly}(|x|)$ (thanks to instance extension property – see Remark 10).
- For any query the vertices V_i^* are always isolated in G_i (if some vertex in V_i^* were to not be isolated, the algorithm could be modified to simply not make the query).
- Exactly t queries are made irrespective of the oracle answers.

We reduce x to a single instance of Lonely$_p$ as follows.

Vertices. The vertices of the Lonely$_p$ instance will be $V = [p]^n \cup [p]^{2n} \cup \cdots \cup [p]^{tn}$, which we interpret as $\overline{V} = V_1 \cup (V_1 \times V_2) \cup (V_1 \times V_2 \times V_3) \cup \cdots \cup (V_1 \times \cdots \times V_t)$. The designated vertices will be $\overline{V}^* := V_1^*$. Note that $|\overline{V}^*| = |V_1^*| \not\equiv 0 \pmod p$.

Edges. We'll define the hyperedge for vertex $\overline{v} = (v_1, \ldots, v_k)$ for any $k \le t$. Let $j \le k$ be the last coordinate such that for all $i < j$, the vertex v_i is a valid solution for the Lonely$_p$ instance (C_i, V_i^*), which the algorithm creates on receiving $v_1, \ldots, v_{i-1}$ as answers to previous queries.

Case $j < k$: Let $u_1, \ldots, u_{p-1}$ be the neighbors of v_k in a canonical trivial matching over $[p]^n$; e.g. $\{[p] \times w : w \in [p]^{n-1}\}$. The neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i$.

Case $j = k$: We consider three cases, depending on whether v_k is designated, non-isolated or isolated in the Lonely$_p$ instance (C_k, V_k^*).

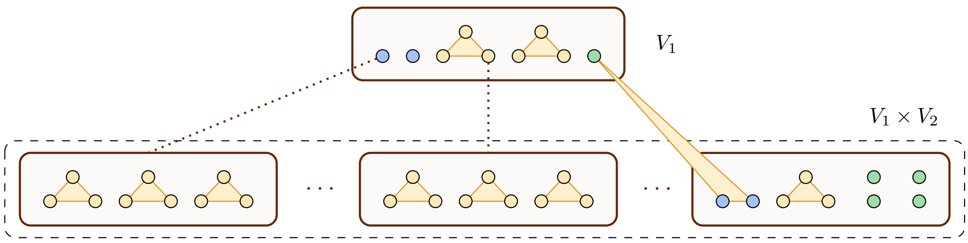
Non-isolated v_k: For $u_1, \ldots, u_{p-1}$ being the neighbors of v_k in G_k, the neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i$.

Isolated v_k: Such a v_k is a valid solution for (C_k, V_k^*).

If $k < t$: the algorithm will have a next oracle query (C_{k+1}, V_{k+1}^*). In this case, for $u_1, \ldots, u_{p-1}$ being the designated vertices in V_{k+1}^*, the neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, v_k, u_i)\}_i$.

If $k = t$: there are no more queries, and we leave $\overline{v}$ isolated.

Designated v_k: Let $u_1, \ldots, u_{p-2}$ be the other designated vertices in V_k^*. The neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i \cup \{(v_1, \ldots, v_{k-1})\}$.



It is easy to see that our definition of edges are consistent and the only vertices which are isolated (apart from those in $\overline{V}^*$) are of the type $(y_1, \ldots, y_t)$ where each y_i is a valid solution for the Lonely$_p$ instance (C_i, V_i^*). Thus, given an isolated vertex $\overline{y}$, we can immediately infer a solution for x as $L(x, y_1, \ldots, y_t)$. This completes the reduction since $\overline{V}$ is efficiently representable and indexable – see Remark 10. ◀

References

1 James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-d tucker is PPA complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:163, 2015. URL: `http://eccc.hpi-web.de/report/2015/163`.

2 Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. `doi:10.1016/0001-8708(87)90055-7`.

3 Noga Alon, Shmuel Friedland, and Gil Kalai. Regular subgraphs of almost regular graphs. *Journal of Combinatorial Theory, Series B*, 37(1):79–91, 1984.

4 Noga Alon and Douglas B. West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986. doi:10.1090/S0002-9939-1986-0861764-9.

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

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

7 Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theor. Comput. Sci.*, 103(1):3–23, 1992. doi:10.1016/0304-3975(92)90084-S.

8 Aleksandrs Belovs, Gábor Ivanyos, Youming Qiao, Miklos Santha, and Siyi Yang. On the polynomial parity argument complexity of the combinatorial nullstellensatz. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 30:1–30:24, 2017. doi:10.4230/LIPIcs.CCC.2017.30.

9 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498. IEEE, 2015.

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

11 Joshua Buresh-Oppenheim. On the TFNP complexity of factoring. *Manuscript*, 2006. URL: http://www.cs.toronto.edu/~bureshop/factor.pdf.

12 Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci.*, 62(2):267–289, 2001. doi:10.1006/jcss.2000.1726.

13 Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Logic*, 163(9):1163–1182, 2012. doi:10.1016/j.apal.2012.01.015.

14 I. Bárány, S. B. Shlosman, and A. Szücs. On a topological generalization of a theorem of tverberg. *Journal of the London Mathematical Society*, s2-23(1):158–164, 1981. doi:10.1112/jlms/s2-23.1.158.

15 Claude Chevalley. Démonstration d'une hypothèse de m. artin. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):73–75, December 1935. doi:10.1007/BF02940714.

16 Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N Rothblum. Finding a nash equilibrium is no easier than breaking fiat-shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1103–1114. ACM, 2019.

17 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652.

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

19 Xiaotie Deng, Jack R. Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-Completeness. In Ran Raz, editor, *31st Conference on Computational Complexity (CCC 2016)*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:25, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.23.

20 Aris Filos-Ratsikas and Paul W. Goldberg. Consensus halving is ppa-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 51–64, 2018. doi:10.1145/3188745.3188880.

21 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *STOC (to appear)*, 2019. arXiv:1805.12559.

22 C. Goldberg and D. West. Bisection of circle colorings. *SIAM Journal on Algebraic Discrete Methods*, 6(1):93–106, 1985. doi:10.1137/0606010.

23 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 38:1–38:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

24 Michelangelo Grigni. A sperner lemma complete for ppa. *Information Processing Letters*, 77(5-6):255–259, 2001.

25 Alexandros Hollender. The classes PPA-k: Existence from arguments modulo k. In Ioannis Caragiannis, Vahab Mirrokni, and Evdokia Nikolova, editors, *Web and Internet Economics*, pages 214–227, Cham, 2019. Springer International Publishing.

26 Emil Jerábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016. doi:10.1016/j.jcss.2015.08.001.

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

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

29 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *Journal of the ACM (JACM)*, 66(5):34, 2019.

30 Pravesh K Kothari and Ruta Mehta. Sum-of-squares meets nash: lower bounds for finding any equilibrium. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1241–1248. ACM, 2018.

31 Donald L. Kreher and Douglas R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*, volume 7 of *Discrete Mathematics and Its Applications*. CRC Press, 1998.

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

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

34 Christian Reiher. On kemnitz'conjecture concerning lattice-points in the plane. *The Ramanujan Journal*, 13(1-3):333–337, 2007.

35 Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 258–265, 2016.

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

37 Ewald Warning. Bemerkung zur vorstehenden arbeit von herrn chevalley. *Abh. Math. Sem. Univ. Hamburg*, 11:76–83, 1936.

A Appendix: Reductions Between Complete Problems

In order to prove Theorem 9, we introduce an additional problem that will serve as intermediate problem in our reductions.

▶ **Definition 47** ($\textsc{Leaf}'_q$).
Principle: *Same as* $\textsc{Leaf}_q$*, but degrees are allowed to be larger (polynomially bounded).*
Object: *q-uniform hypergraph $G = (V, E)$. Designated vertex $v^* \in V$.*
Inputs: ▷ $C : \{0,1\}^n \to (\{0,1\}^{nq})^k$
where $(\{0,1\}^{nq})^k$ is interpreted as k many q-subsets of $\{0,1\}^n$
▷ $v^* \in \{0,1\}^n$ *(usually 0^n)*
Encoding: $V := \{0,1\}^n$.
For distinct $v_1, \dots, v_q$, edge $e := \{v_1, \dots, v_q\} \in E$ if $e \in C(v)$ for all $v \in e$
Solutions: v^* *if* $\deg(v) \equiv 0 \pmod q$ *and*
$v \neq v^*$ *if* $\deg(v) \not\equiv 0 \pmod q$

Proof of Theorem 9. We show the following inter-reducibilities: (1) $\textsc{Leaf}_q \asymp \textsc{Leaf}'_q$, (2) $\textsc{Leaf}'_q \asymp \textsc{Bipartite}_q$ and (3) $\textsc{Leaf}_q \asymp \textsc{Lonely}_q$.

(1a) $\textsc{Leaf}_q \preceq \textsc{Leaf}'_q$. Each instance of $\textsc{Leaf}_q$ is trivially an instance of $\textsc{Leaf}'_q$.

(1b) $\textsc{Leaf}'_q \preceq \textsc{Leaf}_q$. We start with a $\textsc{Leaf}'_q$ instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$ with degree at most k. Let $t = \lceil k/q \rceil$. We construct a $\textsc{Leaf}_q$ instance encoding a hypergraph $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V} := V \times [t]$, intuitively making t copies of each vertex.

In order to locally compute hyperedges, we first fix a canonical algorithm that for any vertex v and any edge $e \in E$ incident on v, assigns it a label $\ell_v(e) \in [t]$, with at most q edges mapping to the same label – e.g. sort all edges incident on v in lexicographic order and bucket them sequentially in at most t groups of at most q each. Note that we can ensure that for any vertex v at most one label gets mapped to by a non-zero, non-q number of edges. Moreover, if $\deg(v) \equiv 0 \pmod q$, then exactly q or 0 edges are assigned to any label.

We'll assume that $\deg(v^*) \not\equiv 0 \pmod q$, as otherwise, a reduction wouldn't be necessary. We let (v^*, ℓ^*) be the designated vertex of the $\textsc{Leaf}_q$ instance, where ℓ^* is the unique label that gets mapped to by a non-zero, non-q number of edges incident on v^*.

For any vertex $(v, i) \in \overline{V}$, we assign it at most q edges as follows: For each edge $e = \{v_1, \dots, v_q\}$ such that $\ell_v(e) = i$, the corresponding hyperedge of (v, i) is

$$(v_1, \ell_{v_1}(e)), \dots, (v_q, \ell_{v_q}(e)) .$$

It is easy to see that the designated vertex (v^*, ℓ^*) indeed has non-zero, non-q degree. Moreover, a vertex $\deg(v, i) \notin \{0, q\}$ in $\overline{G}$ only if v has a non-multiple-of-q degree in G. Thus, solutions to the $\textsc{Leaf}_q$ instance naturally maps to solutions to the original $\textsc{Leaf}'_q$ instance.

By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ is efficiently indexable.

(2a) $\textsc{Leaf}'_q \preceq \textsc{Bipartite}_q$. We start with a $\textsc{Leaf}'_q$ instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$. We construct a $\textsc{Bipartite}_q$ instance encoding a graph $\overline{G} = (\overline{V} \cup \overline{U}, \overline{E})$ such that $\overline{V} = V$ and $\overline{U} = \binom{V}{q}$, i.e. all q-sized subsets of V. We include the edge $(v, e) \in \overline{E}$ if $e \in E$ is incident on v. The designated vertex for the $\textsc{Bipartite}_q$ instance is v^* in $\overline{V}$.

Clearly, all vertices $e \in \overline{U}$ have degree either q or 0. For any $v \in \overline{V}$, the degree of v in $\overline{G}$ is same as its degree in G. Thus, any solution to the $\textsc{Bipartite}_q$ instance immediately gives a solution to the original Leaf'_q instance. By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ and $\overline{U}$ are efficiently indexable (cf. [31, §2.3] for efficiently indexing $\overline{U}$).

(2b) $\text{Bipartite}_q \preceq \text{Leaf}'_q$. We start with a Bipartite_q instance (C, v^*) encoding a bipartite graph $\mathcal{G} = (V \cup U, E)$ with maximum degree of any vertex being at most k. We construct a Leaf'_q instance encoding a hypergraph $\overline{G} = (\overline{V}, \overline{E})$ such that $\overline{V} = V$ with designated vertex v^*.

First, we fix a canonical algorithm that for any vertex $u \in U$ with $\deg_G(u) \equiv 0 \pmod q$ produces a partition of it's neighbors with q vertices of V in each part. Now, the set of q-uniform hyperedges incident on any vertex $v \in \overline{V}$ in $\overline{E}$ can be obtained as: for all neighbors u of v, with $\deg_G(u) \equiv 0 \pmod q$, we include a hyperedge consisting of all vertices in the same partition as v among the neighbors of u (we ignore neighbors u with $\deg(u) \not\equiv 0 \pmod q$).

Observe that $\deg_{\overline{G}}(v) \le \deg_G(v)$ and equality holds if and only if all neighbors of v in G have degree $\equiv 0 \pmod q$. Hence for any $v \in \overline{V}$, if $\deg_{\overline{G}}(v) \neq \deg_G(v) \pmod q$, then there exists a neighbor $u \in U$ of v in G such that $\deg(u) \not\equiv 0 \pmod q$. Thus, if $v = v^*$ and $\deg_{\overline{G}}(v^*) \equiv 0 \pmod q$, then either $\deg_G(v) \equiv 0 \pmod q$ or we can find a neighbor u of v in G with $\deg(u) \not\equiv 0 \pmod q$. Similarly if for some $v \neq v^*$, we have $\deg_{\overline{G}}(v^*) \not\equiv 0 \pmod q$, then either $\deg_G(v) \not\equiv 0 \pmod q$ or we can find a neighbor u of v in G with $\deg(u) \not\equiv 0 \pmod q$. Thus, any solution to the Leaf'_q instance gives us a solution to the original Bipartite_q instance. This completes the reduction since $\overline{V} = \{0,1\}^n$ and the edges are locally computable with black-box access to C.

(3a) $\text{Leaf}_q \preceq \text{Lonely}_q$. We start with a Leaf_q instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$ with degree at most q. If $\deg_G(v^*) = q$ or 0, then we don't need any further reduction. Else, we construct a Lonely_q instance encoding a q-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V} = V \times [q]$. The designated vertices will be $V^* = \{(v, q - i) : 1 \le i \le q - \deg(v^*)\}$. Note that, $|V^*| = q - \deg_G(v^*)$ and hence $1 \le |V^*| \le q - 1$.

In order to locally compute hyperedges, we first fix a canonical algorithm that for any vertex v and any edge $e \in E$ incident on v, assigns it a unique label $\ell_v(e) \in [q]$ – e.g. sort all edges incident on v in lexicographic order and label them sequentially in $[q]$. In fact, we can ensure that an edge incident on v get labeled within $\{1, \dots, \deg_G(v)\}$.

For any vertex $(v, i) \in \overline{V}$, we assign it at most one hyperedge as follows:

- ▷ If $\deg_G(v) = 0$, we include the hyperedge $\{(v, i) : i \in [q]\}$.
- ▷ Else if $\deg_G(v) \ge i$, then for edge $e = \{v_1, \dots, v_q\}$ incident on v such that $\ell_v(e) = i$, the corresponding hyperedge of (v, i) is $(v_1, \ell_{v_1}(e)), \dots, (v_q, \ell_{v_q}(e))$.
- ▷ Else if $0 < \deg_G(v) < i$, we leave it isolated.

It is easy to see that our definition of hyperedges is consistent and that the designated vertices V^* are indeed isolated. Moreover, a vertex (v, i) is isolated in $\overline{G}$ only if $1 \le \deg_G(v) \le q - 1$. Thus, solutions to the Leaf_q instance naturally maps to solutions to the original Leaf'_q instance.

By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ is efficiently indexable.

(3b) $\text{Lonely}_q \preceq \text{Leaf}_q$. We start with a Lonely_q instance (C, V^*), where C encode a q-dimensional matching $G = (V, E)$. We construct a Leaf_q instance encoding a q-uniform hypergraph $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V}$ that will be specified shortly. We describe the hyperedges in $\overline{G}$ and it'll be clear how to compute the hyperedges for any vertex locally with just black-box access to C.

We start with $\overline{V} = V$. Our goal is to transform all vertices of degree 1 to degree q, while ensuring that vertices of degree 0 are mapped to vertices of degree not a multiple of q. Towards this goal we let $\overline{E}$ to be set of edges in E in addition to $q-1$ canonical q-dimensional matchings over V. For example, for a vertex $v := (x_1, \ldots, x_n) \in V = [q]^n$, the corresponding edges in $\overline{E}$ include an edge in E (if any) and edges of the type $e_i = \{(x_1, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_n) : j \in [q]\}$ for $i \in [q-1]$ (note, this requires us to assume $n \geq q-1$). Adding the $q-1$ matchings increases the degree of each vertex by $q-1$. Therefore, vertices with initial degree 1 now have degree q and vertices with initial degree 0 now have degree $q-1$. However, a couple of issues remain in order to complete the reduction, which we handle next.

Multiplicities. An edge $e \in E$ might have gotten added twice, if it belonged to one of the canonical matchings. To avoid this issue altogether, instead of adding edges directly on V, we augment $\overline{V}$ to become $\overline{V} := V \cup \left(\binom{V}{q} \times [q-1]\right)$, i.e. in addition to V, we have $q-1$ vertices for every potential hyperedge of G. For any edge $e := \{v_1, \ldots, v_q\} \in E$, instead of adding it directly in $\overline{G}$, we add hyperedge $\{v, (e,1), (e,2), \ldots, (e, q-1)\}$ for each $v \in e$. Note that, all vertices $(e,i) \in \binom{V}{q} \times [q-1]$ have degree q if $e \in E$ and degree 0 if $e \notin E$, so they are non-solutions for the $\textsc{Leaf}_q$ instance. For vertices in V, we still have as before that vertices with initial degree 1 now have degree q and vertices with initial degree 0 now have degree $q-1$.

Designated vertex. In a $\textsc{Leaf}_q$ instance, we need to specify a single designated vertex $v^* \in \overline{V}$. If the $\textsc{Lonely}_q$ instance had a single designated vertex then we would be done. However, in general it is not possible to assume this (for non-prime q). Nevertheless, we provide a way to get around this. We augment $\overline{V}$ with $t = (q-1)(q-k)+1$ additional vertices to become $\overline{V} := V \cup \left(\binom{V}{q} \times [q-1]\right) \cup \{w_{i,j} : i \in [q-k],\ j \in [q-1]\} \cup \{v^*\}$, where v^* will eventually be the single designated vertex for the $\textsc{Leaf}_q$ instance.

Let $V^* = \{u_1, \ldots, u_k\} \subseteq V$ be the set of designated vertices in the $\textsc{Lonely}_q$ instance (note $1 \leq k < q$). So far, note that $\deg_{\overline{G}}(u_i) = q-1$. The only new hyperedges we add will be among u_i's, $w_{i,j}$'s and v^*, in such a way that $\deg_{\overline{G}}(u_i)$ will become q, the degree of all $w_{i,j}$'s will also be q and degree of v^* will be $q-k$.

- ▷ For each $u \in V^*$, include $\{u, w_{1,1}, \ldots, w_{1,q-1}\}$. So far, $\deg_{\overline{G}}(u) = q$ and $\deg_{\overline{G}}(w_{1,j}) = k$.
- ▷ For each $j \in [q-1]$ and each $i \in \{2, \ldots, q-k\}$, include $\{w_{1,j}, w_{i,1}, \ldots, w_{i,q-1}\}$.
  So far, $\deg_{\overline{G}}(w_{i,j}) = q-1$ for all $(i,j) \in [q-k] \times [q-1]$.
- ▷ Finally, for each $(i,j) \in [q-k] \times [q-1]$, include $\{v^*, w_{i,1}, \ldots, w_{i,q-1}\}$.
  Now, $\deg_{\overline{G}}(w_{i,j}) = q$ for all $(i,j) \in [q-k] \times [q-1]$ and $\deg_{\overline{G}}(v^*) = q-k$.

Thus, we have finally reduced to a $\textsc{Leaf}_q$ instance encoding the graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{V} := V \cup \left(\binom{V}{q} \times [q-1]\right) \cup \{w_{i,j} : i \in [q-k],\ j \in [q-1]\} \cup \{v^*\}$. By Remark 10, this completes the reduction, since $\overline{V}$ is efficiently indexable (again, see [31] for a reference on indexing $\binom{V}{q}$) and the edges are locally computable using black-box access to C. ◀

A.1 Completeness of Succinct Bipartite

We introduce an intermediate problem to show PPA_p–completeness of $\textsc{SuccinctBipartite}_p$.

▶ **Definition 48** ($\textsc{TwoMatchings}_p$).

Principle: *Two p-dimensional matchings over a common vertex set, with a vertex in exactly one of the matchings, has another such vertex.*

Object: *Two p-dimensional matchings $G_0 = (V, E_0)$, $G_1 = (V, E_1)$.*
Designated vertex $v^* \in V$.

Inputs: $\triangleright$ $C_0 : \{0,1\}^n \to (\{0,1\}^n)^p$ *and* $C_1 : \{0,1\}^n \to (\{0,1\}^n)^p$
$\triangleright$ $v^* \in \{0,1\}^n$
Encoding: $V \coloneqq \{0,1\}^n$. *For* $b \in \{0,1\}$, $E_b \coloneqq \{e : C_b(v) = e$ *for all* $v \in e\}$
Solutions: v^* *if* $\deg_{G_0}(v^*) \neq 1$ *or* $\deg_{G_1}(v^*) \neq 0$ *and*
$v \neq v^*$ *if* $\deg_{G_0}(v^*) \neq \deg_{G_1}(v^*)$

Observe that in the case of $p = 2$, TwoMatchings$_p$ can be readily seen as equivalent to Leaf$_2$.

▶ **Theorem 49.** *For any prime p, SuccinctBipartite$_p$ and TwoMatchings$_p$ are* PPA$_p$*–complete.*

Proof. We show Bipartite$_p$ $\preceq$ SuccinctBipartite$_p$ $\preceq$ TwoMatchings$_p$ $\preceq$ Lonely$_p$.

Bipartite$_p$ $\preceq$ SuccinctBipartite$_p$. Since p is a prime, we can assume that the designated vertex v^* has degree $1 \pmod p$ (similar to Lemma 44). Since the number of neighbors in a Bipartite$_p$ instance are polynomial, we can check if an edge exists and canonically group them efficiently for all vertices with degree being a multiple of p. The designated edge e^* is the unique ungrouped edge incident on v^*. Thus, valid solution edges to SuccinctBipartite$_p$ must have at least one endpoint which is a solution to the original Bipartite$_p$ instance.

SuccinctBipartite$_p$ $\preceq$ TwoMatchings$_p$. We reduce to a TwoMatchings$_p$ instance encoding two p-dimensional matchings $\overline{G}_0 = (\overline{V}, \overline{E}_0)$ and $\overline{G}_1 = (\overline{V}, \overline{E}_1)$, over the vertex set $\overline{V} = V \times U \times [p-1]$, that is, all possible edges producible in the SuccinctBipartite$_p$ instance. The designated vertex v^* is the designated edge e^* in the SuccinctBipartite$_p$ instance.

For any edges $e_1, \ldots, e_p$, which are grouped by ϕ_V pivoted at some $v \in V$, we include the hyperedge $\{e_1, \ldots, e_p\}$ in $\overline{E}_0$. Similarly, for any edges $e_1, \ldots, e_p$, which are grouped by ϕ_U pivoted at some $u \in U$, we include the hyperedge $\{e_1, \ldots, e_p\}$ in $\overline{E}_1$. It is easy to see that points in exactly one of the two matchings $\overline{G}_0$ or $\overline{G}_1$ correspond to edges of the SuccinctBipartite$_p$ instance that are not grouped at exactly one end. Thus, we can derive a solution to SuccinctBipartite$_p$ from a solution to TwoMatchings$_p$. (Remark: while edges which are not grouped at either end are solutions to SuccinctBipartite$_p$, they do not correspond to a solution in the TwoMatchings$_p$ instance.)

TwoMatchings$_p$ $\preceq$ Lonely$_p$. Given an instance of TwoMatchings$_p$ that encodes two p-dimensional matchings $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$, we reduce to an instance of Lonely$_p$ encoding a p-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ such that $\overline{V} = V \times [p]$. The designated vertex for the Lonely$_p$ instance is (v^*, p).

For any hyperedge $\{v_1, \ldots, v_p\}$ in E_0, we include the hyperedge $\{(v_1, i), (v_2, i), \ldots, (v_p, i)\}$ in $\overline{G}$ for each $i \in \{1, \ldots, p-1\}$. Similarly, for any hyperedge $\{v_1, \ldots, v_p\}$ in E_1, we include the hyperedge $\{(v_1, p), (v_2, p), \ldots, (v_p, p)\}$ in $\overline{G}$. If $v \in V$ is isolated in both G_0 and G_1, then we include the hyperedge $\{v\} \times [p]$.

Observe that, (v^*, p) is isolated by design. A vertex (v, i), for $i < p$ is isolated only if $\deg_{G_0}(v) = 0$ and $\deg(G_1) = 1$. Similarly, the vertex (v, p) is isolated only if $\deg_{G_0}(v) = 1$ and $\deg(G_1) = 0$. Thus, isolated vertices in the Lonely$_p$ instance correspond to solutions of the TwoMatchings$_p$ instance. ◀

A.2 Equivalence with PMOD$_p$

Proof of Lemma 44. Consider any prime p. Consider a Lonely$_p$ instance (C, V^*), where C encodes a p-dimensional matching $G = (V, E)$ and $|V^*| = \ell$. We wish to reduce to an instance of Lonely$_p^k$, where the number of designated vertices is exactly k. First, we'll assume that all vertices in V^* are indeed isolated in G, otherwise, no reduction would be necessary. The key reason why this lemma holds for primes (and not for composites) is because ℓ has a multiplicative inverse modulo p. In particular, let $t \equiv \ell^{-1}k \pmod p$.

We construct a $\textsc{Lonely}_p^k$ instance encoding the p-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ over $\overline{V} = V \times [t]$. We let $\overline{V}^*$ to be the lexicographically first k vertices in $V^* \times [t]$. Note that $|V^* \times [t]| = t.\ell \equiv k \pmod p$. Thus, we partition the remaining vertices of $V^* \times [t]$ into p-uniform hyperedges. For any vertex $v \in V \smallsetminus V^*$, with neighbors $v_1, \ldots, v_{p-1}$ in G, the neighbors of (v, i) in $\overline{G}$ are $(v_1, i), \ldots, (v_{p-1}, i)$ for any $i \in [t]$. Thus, a vertex (v, i) is isolated only if it is in $\overline{V}^*$ or v is isolated in G. This completes the reduction since $\overline{V}$ is efficiently indexable – see Remark 10. ◀

Proof of Corollary 45. It is easy to see that $\textsc{Mod}_q \le \textsc{Lonely}_q$ with number of designated vertices being $k \equiv -2^n \pmod q$, since $\{0,1\}^n$ is efficiently indexable (Remark 10). Conversely, using Lemma 44, we can reduce a $\textsc{Lonely}_q$ instance to a $\textsc{Mod}_q$ instance as follows: Let the $\textsc{Lonely}_q$ instance encode a q-dimensional matching over $[q]^n$ with k designated vertices. If any of the designated vertices are not isolated, no further reduction is necessary. Otherwise, we can embed the non-designated vertices of G into the first $q^n - k$ vertices of $\{0,1\}^N$ for a choice of N satisfying $2^N > q^n$ and $2^N \equiv -k \pmod q$. Such an N is guaranteed to exist (and can be efficiently found) when q is a prime. Since $2^N - q^n + k \equiv 0 \pmod q$, we can partition the remaining vertices into q-uniform hyperedges, and thus, solutions to the $\textsc{Mod}_q$ instance readily map to solutions of the original $\textsc{Lonely}'_q$ instance. ◀

B Appendix: Proof of Theorem 36

Proof of Theorem 36. We show a reduction from $\textsc{ChevalleyWithSymmetry}_p$ to $\textsc{SuccinctBipartite}_p[\mathsf{AC}^0_{\mathbb{F}_p}]$; the theorem then follows by combining this reduction with Theorem 3. Additionally from the proof of Theorem 3 we can assume without loss of generality that the system of polynomials $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$ of the $\textsc{ChevalleyWithSymmetry}_p$ instance has the following properties.

a. Each polynomial f_i has degree at most 2.
b. Each polynomial f_i has at most 3 monomials.
c. Each polynomial f_i has at most 3 variables.

Hence, we can compute each of the polynomials g_i^{p-1} explicitly as a sum of monomials. The degree of this polynomial is $O(p)$ and the number of monomials is at most 3^p. Observe that since p is a constant, 3^p is also a constant.

Now we follow the proof of Lemma 25 that reduces $\textsc{ChevalleyWithSymmetry}_p$ to $\textsc{SuccinctBipartite}_p$. Following this proof there are two circuits that we need to replace with formulas in $\mathsf{AC}^0_{\mathbb{F}_p}$ to reduce to $\textsc{SuccinctBipartite}_p$. The first circuit is the edge counting circuit $\mathcal{C}$ and the second is the grouping function ϕ. We remind that the bipartite graph $G(U, V)$ of the $\textsc{SuccinctBipartite}_p$ instance has two parts U, V, where U is the set of all possible assignments, i.e. $\mathbb{F}_p^n$, and $V = V_1 \cup V_2$, where V_1 in the set of all monomials of the polynomial $F = \prod_{i=1}^m (1 - g_i^{p-1})$ and V_2 is the set of all p-tuples of assignments, i.e. $(\mathbb{F}_p^n)^p$.

From Edge Counting Circuit To Edge Counting Formula. As described in the proof of Lemma 25 the edge counting circuit takes as input a vertex $u \in U$ and a vertex $v \in V$ and outputs the multiplicity of the edge $\{u, v\}$ in G. Hence, the edge counting formula $\mathcal{C}$, that we want to implement, takes as input a tuple $(\boldsymbol{x}, s, \boldsymbol{a}, \boldsymbol{y})$. The vector $\boldsymbol{x}$ corresponds to the assignment in U. The vector $\boldsymbol{a}$ corresponds to the description of a monomial of F, as the product $\prod_{i=1}^m t'_{ia_i}$ where t'_{ia_i} is the a_i-th monomial of the polynomial $1 - g_i^{p-1}$. The vector

$\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p)$ and corresponds to a p-tuple in V_2. Finally, s is a selector number to distinguish between $v \in V_1$ and $v \in V_2$, namely if $s = 1$, we have $v \in V_1$ and if $s = 0$, we have that $v \in V_2$. So, the edge counting formula can be written as follows

$$\mathcal{C}(\boldsymbol{x}, s, \boldsymbol{a}, \boldsymbol{y}) = \left(\prod_{i \in \mathbb{F}_p, i \neq 1} (s - i) \right) \mathcal{C}_1(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}) + \left(\prod_{i \in \mathbb{F}_p, i \neq 0} (s - i) \right) \mathcal{C}_2(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}). \tag{B.1}$$

This way we can define the edge counting formula $\mathcal{C}_1$ for when $v \in V_1$ and the edge counting formula $\mathcal{C}_2$ for when $v \in V_2$ separately and combine them by using at most two additional layers in the arithmetic formula. Now, $\mathcal{C}_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) = \mathbb{1}(\boldsymbol{y} = \boldsymbol{0}) \cdot \prod_{i=1}^{m} \mathcal{Q}_i(\boldsymbol{x}, a_i)$ where $\mathcal{Q}_i(\boldsymbol{x}, a_i)$ is the formula to compute the value $t_{i,a_i}(\boldsymbol{x})$. Observe that the factor $\mathbb{1}(\boldsymbol{y} = \boldsymbol{0})$ can be easily computed and is necessary since $\mathcal{C}_1$ should consider only neighbors between $\boldsymbol{x}$ and monomials in V_1. Hence, if $\boldsymbol{y}$ is not equal to $\boldsymbol{0}$, $\mathcal{C}_1$ should return 0. As we already explained the number of monomials of $1 - g_i^{p-1}$ is constant, and hence the formula $\mathcal{Q}_i(\boldsymbol{x}, a_i)$ can be easily implemented in constant depth using a selector between all different monomials similarly to Equation (B.1). Hence, $\mathcal{C}_1$ is implemented in constant depth.

The formula $\mathcal{C}_2$ has a factor $\mathbb{1}(\boldsymbol{a} = 0)$ to ensure only neighbors in V_2 have non-zero outputs. The main challenge in the description of $\mathcal{C}_2$ is that every distinct p-tuple $\boldsymbol{y}$ has $p!$ equivalent representations, but the modulo p argument of Lemma 25 applies only when edges appear to precisely one of the equivalent copies of the p-tuple. Thus, we let $\mathcal{C}_2$ add edges only to the lexicographically ordered version of $\boldsymbol{y}$. It is a simple exercise to see that sorting of $p!$ numbers, when p is constant, is possible in constant depth. We leave this folklore observation as an exercise to the reader. Once we make sure that $\boldsymbol{y}$ is lexicographically sorted, we compute a sorted representation of the set $\Sigma_{\boldsymbol{x}} = \{\boldsymbol{x}, \sigma(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})\}$, where σ is the permutation in the input of the ChevalleyWithSymmetry$_p$ problem. Then, we can easily check whether the p-tuple represented by $\boldsymbol{y}$ is the same as the sorted p-tuple $\Sigma_{\boldsymbol{x}}$. Finally, we observe that edges between $\boldsymbol{x}$ and $\Sigma_{\boldsymbol{x}}$ are only used when $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}}_{\boldsymbol{h}}$ which again can be checked with constant depth formulas. If these checks pass, then $\mathcal{C}_2$ outputs $p - 1$, otherwise it outputs 0.

From Grouping Circuit to Grouping Formula. For this step we use selectors similarly to Equation (B.1) and sorting as in the description of $\mathcal{C}_2$. We consider two different cases for the grouping formula ϕ. When the first argument is in U, i.e. grouping with respect to an assignment, we call the formula ψ and when the first argument is in V, i.e. grouping with respect to monomials/p-tuples, we call the formula χ. Then, ϕ selects between ψ and χ using a selector. This adds at most two layers to ϕ.

Grouping formula for $x \in U$. First, we describe ψ with inputs $\boldsymbol{x} \in U$, $(s, \boldsymbol{a}, \boldsymbol{y}) \in V$ and r be the copy of the input edge. We have two cases with respect to whether $s = 1$ or $s = 0$. Let ψ^1 be the formula for the first case and ψ^2 be the formula for the second case. For the case $s = 1$, we need again to consider two cases: (i) $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{g}}$ and (ii) $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$. For case (i) we describe the formula ψ_1^1 and for case (ii) we define the formula ψ_2^1. It is easy to see that computing $\mathbb{1}(\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}})$ can be done using a depth 3 formula since $\boldsymbol{g}$ is given in an explicit form. Hence, once again, we can combine ψ_1^1 and ψ_2^1 using a selectors.

Case $s = 1$, $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{g}}$. The formula ψ_1^1 first computes $i^\star = \min_{i : 1 - g_i^{p-1}(\boldsymbol{x}) = 0} i$. This is doable in constant depth, since we can compute in parallel the value $\mathbb{1}(1 - g_i^{p-1}(\boldsymbol{x}) = 0)$ for all $i \in [m_1]$ and then in an extra layer compute for every i whether $1 - g_i^{p-1}(\boldsymbol{x}) = 0$ and $1 - g_j^{p-1}(\boldsymbol{x}) \neq 0$ for all $j < i$, which requires just one multiplication gate per i.

Next, we define a formula ψ^1_{1i} for all i and we use a selector to output $\psi^1_{1i^*}$. In ψ^1_{1i}, we first compute the value $C_i(\boldsymbol{x}) = \prod_{j \neq i} t_{j,a_j}(\boldsymbol{x})$. The output of ψ^1_{1i} is a p-tuple, where each of the p parts differs only on the coordinate a_i of $\boldsymbol{a}$, which corresponds to a monomial of $1 - g_i^{p-1}$, and the value r. We need to determine p different values for the tuple (a_i, r) where $a_i \in [3^p]$, $r \in \mathbb{Z}_p$. These values only depend on the evaluation of the polynomial g_i on the input $\boldsymbol{x}$, on the value a_i and on the value r.
Because of the properties of the input system of polynomials $\boldsymbol{f}$, each polynomial g_i depends only on three variables in $\mathbb{Z}_p$, let these variables be x_1, x_2, x_3 for simplicity. Then, for every i the grouping function that we want to implement is a function with input domain $\mathbb{Z}_p^3 \times [3^p] \times \mathbb{Z}_p$ and output domain $\mathbb{Z}_p^2$. The truth-table of this function has size that depends only on p and therefore we can explicitly implement this function using its truth-table in constant depth. This finishes the construction of ψ^1_{1i}.

Case $\boldsymbol{s = 1}$, $\boldsymbol{x} \in \boldsymbol{\mathcal{V}_g}$. We remind that $\boldsymbol{a} = \boldsymbol{0}$ corresponds to the constant monomial 1 of the polynomial F. If $\boldsymbol{a} \neq \boldsymbol{0}$, this case is similar to the previous, except that we use the polynomials $\boldsymbol{g}_i^{p-1}$ instead of $1 - \boldsymbol{g}_i^{p-1}$, see also the proof of Lemma 25. If $\boldsymbol{a} = \boldsymbol{0}$, ψ^1_2 outputs the input edge $(1, \boldsymbol{a}, \boldsymbol{0}, 1)$ and $p-1$ edges of the form $(0, \boldsymbol{0}, \boldsymbol{y}, t), t \in [p-1]$ where $\boldsymbol{y}$ is the lexicographically ordered set $\Sigma_{\boldsymbol{x}}$.

Case $\boldsymbol{s = 0}$. In this case, the formula ψ^2 checks whether the vector $\boldsymbol{y}$ is in lexicographic order as described in the edge counting formula $\mathcal{C}$ and $\boldsymbol{a} = \boldsymbol{0}$. It also checks if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}_1} \cap \overline{\mathcal{V}}_{\boldsymbol{f}_2}$ as described before. If any of these checks fails, the output is $\boldsymbol{0}$. Otherwise, if $\boldsymbol{y} = \Sigma_{\boldsymbol{x}}$, then we output $p-1$ copies of the edge $(0, \boldsymbol{0}, \boldsymbol{y}, t), t \in [p-1]$, that connects $\boldsymbol{x}$ with $\boldsymbol{y}$, and the edge $(1, \boldsymbol{0}, \boldsymbol{0}, 1)$, that connects $\boldsymbol{x}$ with the constant term of F.

Grouping formula for vertices in V. We describe the grouping formula χ when the first argument belongs to V, i.e. the grouping with respect to monomials or p-tuples. The input again is a triple $(s, \boldsymbol{a}, \boldsymbol{y})$ representing a vertex in V, a vertex $\boldsymbol{x} \in U$ and a number $r \in \mathbb{Z}_p$ that denotes the index of the edge that we want to group, among its possible multiple copies. Again we have two cases, $s = 1$ and $s = 0$, which correspond to the formulas χ^1 and χ^2 respectively. In each case, we have to check that one of $\boldsymbol{a}$, $\boldsymbol{y}$ is equal to $\boldsymbol{0}$, which is done similarly to the previous formulas.

Case $\boldsymbol{s = 1}$. In this case, the input is a monomial $t_{\boldsymbol{a}}(\boldsymbol{x}) = \prod_{i=1}^{m_1} t_{i,a_i}(\boldsymbol{x})$ and we have to find a variable that appears with degree less than $p-1$. We first construct a formula χ^1_j that computes z^k, where k is the degree of x_j in $t_{\boldsymbol{a}}(\boldsymbol{x})$. This can be done with a constant size formula that for a given index j multiplies the powers of x_j in the monomials of $1 - g_i^{p-1}$ appearing in t.
Now, we compute all values $\chi^1_j(1), \ldots, \chi^1_j(p-1)$ and we check in parallel if at least one of them is different from 1. If this is the case, then the degree of x_j in $t(\boldsymbol{x})$ is less than $p-1$. Hence, we have computed the formula $\bar{\chi}^1_j(\boldsymbol{a}) = \mathbb{1}(\text{degree of } x_j \text{ in } t_{\boldsymbol{a}} \neq p-1)$. We can find the smallest index j^* such that $\bar{\chi}^1_j(\boldsymbol{a}) = 1$ using the same construction as in ψ^1. So, we can construct a formula for each j that is equal to 1 if and only if $j = j^*$ is the smallest index such that x_{j^*} has degree less than $p-1$ in $t_{\boldsymbol{a}}$. Finally, we use a selector to find the value $C_{j^*}(\boldsymbol{x}) = x_{j^*}^{-k} t(\boldsymbol{x})$, by computing $C_j(\boldsymbol{x})$ for all j. This is done through the product of all variables that appear in $t_{\boldsymbol{a}}(\boldsymbol{x})$ excluding x_j.
It is left to implement a formula that takes as input the value $C_{j^*}(\boldsymbol{x}) \in \mathbb{Z}_p$, the value of $r \in \mathbb{Z}_p$ and the values $\chi^1_{j^*}(0), \chi^1_{j^*}(1), \ldots, \chi^1_{j^*}(p-1)$ all in $\mathbb{Z}_p$ and outputs a group of p values in $\mathbb{Z}_p^2$, which corresponds to the values of x_j and r in the output. Observe that both the input and the output size of this formula are only a function of p and, hence, constant. Therefore, we can explicitly construct a constant depth formula to capture this grouping.

Case $s = 0$. For constructing the formula χ^2 we first check whether $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{f}_1}$ and whether $\boldsymbol{y}$ is the lexicographically sorted version of $\Sigma_{\boldsymbol{x}}$. These can both be done as we have described in the construction of the formula ψ above. If all checks pass, then we output the p edges of the form $(\boldsymbol{z}, r)$ for all $\boldsymbol{z} \in \Sigma_{\boldsymbol{x}}$, that correspond to the r-th copy of the edge between $\boldsymbol{z}$ and $\boldsymbol{y}$.

Combining the formulas ψ and χ through a selector concludes the construction of ϕ. Hence, the theorem follows by observing that the instance of ChevalleyWithSymmetry$_p$ that we get when reducing Lonely$_p$ to ChevalleyWithSymmetry$_p$ in Theorem 3 reduces to SuccinctBipartite$_p$[$\mathsf{AC}^0_{\mathbb{F}_p}$]. ◀