Multivariate polynomials modulo prime powers: their roots, zeta-function and applications.

A thesis submitted
in Partial Fulfillment of the Requirements
for the Degree of

Dual BT-MT

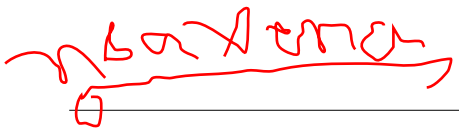
by

Sayak Chakrabarti
17807648

to the
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
May, 2022

CERTIFICATE

It is certified that the work contained in the thesis titled **Multivariate polynomials modulo prime powers: their roots, zeta-function and applications.**, by **Sayak Chakrabarti**, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Prof. Nitin Saxena

Department of Computer Science & Engineering

IIT Kanpur

May, 2022

Declaration

This is to certify that the thesis thesis titled **Multivariate polynomials modulo prime powers: their roots, zeta-function and applications.**, by **Sayak Chakrabarti**, has been authored by me. It presents the research conducted by me under the supervision of **Prof. Nitin Saxena**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations (if any) with appropriate citations and acknowledgements, in line with established norms and practices.

Signature

Name: Sayak Chakrabarti

Program: Dual BT-MT

Computer Science and Engineering,

Indian Institute of Technology Kanpur,

Kanpur- 208016

May, 2022

Place: Kanpur, India

ABSTRACT

Name of student: **Sayak Chakrabarti** Roll no: **17807648**

Degree for which submitted: **Dual BT-MT**

Department: **Computer Science & Engineering**

Thesis title: **Multivariate polynomials modulo prime powers: their roots, zeta-function and applications.**

Name of Thesis Supervisor: **Prof. Nitin Saxena**

Month and year of thesis submission: **May, 2022**

Finding and counting roots of polynomials in prime power rings are two fundamental problems that have far reaching applications in mathematics and computer science. This problem has been solved for univariates while it was open for multivariates as they presented challenges of their own.

These problems of polynomials modulo prime powers have been of interest among mathematicians for a long time, and have been widely used ever since Hensel 1918 gave the famous Hensel's lifting. We use the machinery developed in Berthomieu et al. 2013 to reduce these problems modulo p^k to $\mathbb{F}_p$ and solve them using some new techniques and observations.

In this thesis, we explore these two problems of root finding and root counting by giving different algorithms. Formally speaking, given a polynomial $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$, we can efficiently represent using a datastructure, and thereby find and count, all of its roots modulo p^k, for a prime p and an integer $k \geq 2$. However, this algorithm works for small primes when the degree d and number of variables n of $f(\mathbf{x})$ are constant. As an application to this, we can describe all the roots of $f(\mathbf{x})$ over $\mathbb{Z}_p$ as well, thereby leading to an efficient computation and proof of rationality of the Igusa's Local Zeta Function. Following a similar technique, we find a common root

of the system of n-variate polynomials $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ of degrees at most d modulo p^k where n and k are constants. This has interesting applications to finding certain factors of an univariate polynomial $f(x)$ mod p^k.

Acknowledgements

Pursuing the degree at IITK and working on my thesis has been quite a journey, and I greatly indebted to several people who have helped me complete it. First and foremost, I would like to convey my sincere gratitude towards my advisor, Prof. Nitin Saxena, for his guidance, support and patience throughout the span of 1.5 years. Apart from introducing me to beautiful concepts in mathematics, he has also taught me how to understand seemingly complicated mathematical theories in terms of 'computer science definitions'. I have always been amazed at his intuition on problems, and have learned a lot from him on how to start thinking as a researcher. I am thankful for all the education he has imparted on me under his watchful eyes, never losing patience even when our discussions went on for hours, be it during office times or as late as 2-3 am in the morning, and always making sure that I have followed the concepts. I feel fortunate to have had him as my advisor, as well as thankful to have learned about theoretical CS from the four courses I took under him.

I'd like to extend my heartfelt gratitude to Prof. Rajat Mittal for giving me a chance to work under his guidance in my third year, when I really had no idea of TCS. Looking back, I remember all the silly doubts I would ask him and at every meeting he would always make sure that I have understood everything, however insignificant it may seem, before moving on to newer stuff. Without his influence, I probably would never have known the beauty of TCS and I feel lucky to have worked with him.

I feel really fortunate to have worked with Prof. Petteri Kaski and hope to have

picked up some of the wisdom he has shared with me. He has taught me how to think independently like a researcher and have always encouraged me to explore new things of my own. Working with him has always been entertaining and intriguing along his poetic metaphors with mathematics. I would also like to thank him for putting up with discussions with me remotely despite the hurdles and providing insightful remarks at every step, to help me grow and learn more about research.

This thesis and my general interest in TCS would not have carried forward if it were not for them. I'd also like to thank my coauthors, Ashish, for all the discussions and advice during researchwork, and the suggestions to improve this thesis. Friends and family have had a huge impact over the years, especially when the pandemic (and the neverending boredom) had made concentrating much more difficult. I would like to thank my labmates Devansh, Diptajit, Sagnik and Sanyam for the wonderful moments as well as entertaining discussions including but not restricted to research. Staying at home for 2.5 years had been tough, and it would not have been possible to overcome the long period of time without the support and company of my friends at IITK and Kolkata.. I am sorry I can not mention all of them as the list would have been unending, but everyone has meant a lot to me.

Last but not the least, I would like to thank my parents for all the unconditional love and support through the years, especially during the pandemic in order to make remote working as bearable as possible, and my sister and cousins for completely preventing that from happening. I've grown to enjoy research in TCS and none of it would not have been possible without their sacrifice and their constant push, which is why I am dedicating this thesis to them.

To my parents,

Contents

List of Publications

[CDS22] **Factoring modular polynomials via Hilbert's Nullstellensatz**
Sayak Chakrabarti, Ashish Dwivedi and Nitin Saxena
submitted, 2022.

[CS22] **Describing the roots of multivariates** $\mathrm{mod} p^k$ **and efficient computation of Igusa's Local Zeta Function**
Sayak Chakrabarti and Nitin Saxena
submitted, 2022.

Chapters 3 and 4 focus on [CS22], while Chapters 5 and 6 focus on [CDS22].

Chapter 1

Introduction

We address two important problems in computational algebra– describing the all the roots of multivariates and finding a common solution to a system of polynomial equations modulo prime powers.

Describing roots of multivariates. The first problem of describing the roots is: given an n-variate polynomial $f(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ of degree at most d, we give an algorithm and an efficient data-structure to return all the roots of $f(\mathbf{x}) \equiv 0 \bmod p^k$, for a prime p and an integer $k \geq 2$. The result can be formally stated as follows.

Theorem 1.1 (Describing the roots)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[\mathbf{x}]$ of degree d and n variables, we can efficiently describe all the roots of $f(\mathbf{x}) \equiv 0 \bmod p^k$ in deterministic $\mathsf{poly}(k^d, p^d, d^d)$ time, where $d, \log p$ are small.*

Notice that $\mathbb{Z}/p^k\mathbb{Z}$ is not a unique factorization domain, implying that even univariate polynomials can have several roots. There have been some previous works by [Pan95; BLQ13; NRS17; DMS21] where they give algorithms and data-structures, called *representative roots*, to return all the roots of univariates $\bmod p^k$. Their main idea is to reduce the problem to finding each p-adic coordinate by considering a polynomial over $\mathbb{F}_p$, a field where we can 'nicely' find roots. We will specifically refer to Berthomieu et al. [BLQ13] and extend their method to multivariates for lifting roots from $\mathbb{F}_p$ to $\mathbb{Z}/p^k\mathbb{Z}$

However, when we move on to multivariates, several obstacles come into place. The previous works base their analysis on finding each p-adic coordinate by finding roots of a 'lifted' polynomial over $\mathbb{F}_p$. However, for multivariates, even $O(p)$-many roots can exist, which can lead to a large blowup in the time complexity. Furthermore, roots of multivariates do not correspond to factors, as is the case in univariates. It might also be noteworthy that properties of univariates differ vastly from bivariates, while bivariates and general multivariates have similar properties [Kal85; KT90].

In Chapter 3, we address this problem, by first solving for bivariates and then extending the idea to n-variates, for any constant n. The main idea is to reduce the problem of finding roots of the n-variate polynomial to finding common roots of a system of $(n-1)$-variate polynomials. In this process, we observe several nice properties of the structure of the polynomial and give the complete algorithm to return all of its roots $\mathrm{mod} p^k$.

Since we are able to describe all the roots, we can also use this technique to finding roots of $f(\mathbf{x})$ over $\mathbb{Z}_p$, as $\mathbb{Z}/p^k\mathbb{Z}$ behaves quite similar to $\mathbb{Z}_p$ for 'large' k's. Since the behavior of roots modulo all prime powers are given by this, we are also able to compute the Igusa's Local Zeta Function (LZF) [Igu74; Igu77; Igu07], which also proves the rationality of the Poincaré series, in a proof much simpler than that of Denef [Den84]. These problems have been addressed in Chapter 4.

Finding a root of a polynomial system. Given m polynomials $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ in n-variables of degree at most d, we find a common root of the system modulo p^k. In fact, we solve this for a slightly more general ring, of the form $\mathbb{Z}[z]/\langle \varphi(z), p^k \rangle$, for a polynomial $\varphi(z)$ irreducible over $\mathbb{F}_p$, which are known as Galois rings. As will be seen in Chapter 4, $\mathbb{Z}/p^k\mathbb{Z}$ and $\mathbb{Z}_p$ have similar properties for large k. Thus, our main focus here are small values of k, and we also assume n to be constant. We further solve this for a slightly more general ring $\mathbb{Z}[z]/\langle \varphi(z)p^k \rangle$ instead of $\mathbb{Z}/p^k\mathbb{Z}$. The following theorem states the problem.

Theorem 1.2 (HN_{p^k})**.** *Let $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{Z}[z][\mathbf{x}]$ be a set of n-variate degree d polynomials, where n is a constant. Given a prime power p^k for constant k and an $\mathbb{F}_p$-irreducible polynomial $\varphi(z)$, we can efficiently find a common root of the system $f_i(\mathbf{x}) \equiv 0 \bmod \langle \varphi(z), p^k \rangle$, for $i \in [m]$, in randomized $\mathsf{poly}(d^{c_n k}, m, \deg(\varphi), \log p)$ time.*

The complete algorithm has been described in Chapter 5.

Dwivedi et al. [DMS21] reduced the problem of factoring modulo small prime powers to finding roots of a system of multivariates over a Galois ring. We solve this problem by using our technique of polynomial system solving, which we will refer to as Hilbert's Nullstellensatz modulo p^k, HN_{p^k}; and finding roots of the system– thereby giving certain factors. We cite the techniques for this in Chapter 6.

Chapter 2

Preliminaries

2.1 Notations

We use $\mathbf{x}$ to denote the tuple $(x_1, x_2, \ldots, x_n)$. Operations are similarly defined as $\mathbf{a} + \mathbf{b} := (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$, and $c \cdot \mathbf{a} := (c \cdot a_1, \ldots, c \cdot a_n)$, for a scalar c. Similarly, for $\mathbf{i} = (i_1, i_2, \ldots, i_n)$, we have $\mathbf{x}^{\mathbf{i}} = x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$ with degree $|\mathbf{i}|$, and $\mathbf{i}! := i_1! i_2! \ldots i_n!$.

Based on the Taylor's expansion of polynomials in univariates, we define multivariate Taylor's expansion.

Definition 2.1 (Taylor's expansion/ series)**.** *Given a polynomial $f(\mathbf{x})$ of degree d, we can write it as (over any characteristic)*

$$f(\mathbf{a}+\mathbf{x}) \;=\; \sum_{\ell=0}^{\infty} \left(\sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^{\mathbf{i}}} f(\mathbf{a})}{\mathbf{i}!} \cdot \prod_{j=1}^{n} x_j^{i_j} \right), \tag{2.1}$$

where $\partial_{\mathbf{x}^{\mathbf{i}}} f := \frac{\partial^{i_1+\cdots+i_n} f}{\partial x_1^{i_1} \ldots \partial x_n^{i_n}}$ is an order $|\mathbf{i}|$ *partial derivative.*

For a prime p, we can write any integer a as a power series $a =: a_0 + a_1 p + a_2 p^2 + \ldots$, for $a_i \in \{0, 1, \ldots, p-1\}$. We write $\tilde{a} \in \mathbb{Z}_p$, *the ring of p-adic integers*, as a tuple $(a_0, a_1, a_2, \ldots)$. The j-th coordinate corresponds to a_j, and $\tilde{a} \bmod p^k$ is defined as the projection upto the $(k-1)$-th coordinate, i.e. $a_0 + a_1 p + \ldots a_{k-1} p^{k-1}$. Similarly, we define *the field of p-adic numbers* as the fraction field of $\mathbb{Z}_p$, denoted as $\mathbb{Q}_p$. For

more literature on p-adic numbers, we direct the reader to [Gou97; Kob12].

Definition 2.2 (Valuation)**.** *For an integer n and a prime p, we define its* valuation *w.r.t. p, denoted $v_p(n)$, as the largest integer v such that $p^v | n$.*

We now define *Galois rings*, a special kind of rings which have properties similar to that of finite fields, but have composite characteristics.

Definition 2.3 (Galois rings [McD74])**.** *A Galois ring $G(p^k, b)$ is the ring $\mathbb{G} := \mathbb{Z}[z]/\langle p^k, \varphi(z)\rangle$, where p^k is a prime-power and $\varphi(z) \in \mathbb{Z}[z]$ is a polynomial of degree b which is irreducible modulo p.*

It can be directly seen that $\mathbb{G}$ has characteristic p^k and size p^{kb}. Further properties of symmetries, uniqueness etc. can be found in [DMS19; LN94; McD74].

2.2 Basic algebra and algebraic geometry

2.2.1 Resultants

Theorem 2.4. *Let $f(x), g(x) \in \mathbb{Z}[x]$. f and g have a common factor of degree greater than or equal to 1 if and only if there exists polynomials $A(x), B(x)$ satisfying the conditions:*

1. $A, B \neq 0$
2. $\deg(A) < \deg(g)$ *and* $\deg(B) < \deg(f)$
3. $A.f + B.g = 0$

Proof. Suppose f, g have a common factor h, $\deg(h) \geq 1$ and $f = f_1 h, g = g_1 h$. Consider the polynomials A, B as $A = g_1, B = -f_1$. It can be verified that all the three conditions are satisfied.

For the converse direction, we are given with polynomials A, B such that the three conditions are satisfied. Let us assume that the gcd of f, g is 1. By Extended Euclidean Algorithm we can compute polynomials A', B' with $\deg(B') < \deg(f)$,

$\deg(B') < \deg(g)$ such that $A'f + B'g = 1$. Since $Af + Bg = 0$, multiplying the previous equation by B we get

$$B = B(A'f + B'g) = A'Bf + B'(Bg) = (A'B - AB')f,$$

which violates the condition that $B \neq 0$ and $\deg(B) < f$. □

Based on this result, we give an algorithm to check if f and g have common roots by reducing the problem to that of finding a solution to a set of simultaneous equations. Let $A = c_0 + c_1x + \ldots c_{m-1}x^{m-1}$ and $B = d_0 + d_1x + \ldots d_{l-1}x^{l-1}$. We also write $f = a_0 + a_1x + \ldots a_lx^l$ and $g = b_0 + b_1 + \ldots b_mx^m$. Substituting accordingly we get the set of simultaneous equations:

$$\begin{array}{lllll}
a_0c_0 & + & b_0d_0 & = 0 & \text{Coefficient of } x^{l+m-1} \\
a_1c_0 + a_0c_1 & + & b_1d_0 + b_0d_1 = 0 & & \text{Coefficient of } x^{l+m-2} \\
\vdots & & \vdots & \vdots & \vdots \\
a_lc_{m-1} & + & b_md_{l-1} & = 0 & \text{Coefficient of } x^0
\end{array}$$

Definition 2.5 (Sylvester matrix)**.** *Given polynomials f, g, notation as above, the Sylvester matrix of f and g is the coefficient matrix of the set of the above system of equations. Let us denote Sylvester matrix of f, g as $S(f, g)$ by the following $(l + m) \times (l + m)$ matrix:*

$$\begin{pmatrix}
a_l & & & & b_m & & \\
a_{l-1} & \ddots & & & b_{m-1} & \ddots & \\
\vdots & & \ddots & & \vdots & & b_m \\
& \ddots & & a_{l-1} & b_0 & & \vdots \\
& & \ddots & \vdots & & \ddots & \vdots \\
& & & a_0 & & & b_0
\end{pmatrix}$$

Definition 2.6 (Resultant)**.** *The resultant of polynomials f, g denoted by* $\mathsf{Res}(f, g)$ *is the determinant of their Sylvester matrix, i.e.* $\mathsf{Res}(f, g) = \det(S(f, g))$.

Lemma 2.7. *Given two polynomials $f, g \in \mathbb{Z}[x]$, f and g have a common factor of degree greater than or equal to 1 if and only if* $\mathsf{Res}(f, g) = 0$.

Theorem 2.8. *Given $f, g \in \mathbb{Z}[x]$, $\exists A, B \in \mathbb{Z}[x]$ with degrees less than those of g and f respectively, such that* $Af + Bg = \mathsf{Res}(f, g)$.

Proof. If f, g have a common factor, then resultant is 0. Suppose they do not have a common factor, we have A', B' with same degree constraints such that $A'f + B'g = 1$. Now from the equations:

$$\begin{array}{lllll}
a_0c_0 & + & b_0d_0 & = 0 & \text{Coefficient of } x^{l+m-1} \\
a_1c_0 + a_0c_1 & + & b_1d_0 + b_0d_1 = 0 & & \text{Coefficient of } x^{l+m-2} \\
\ddots & & \ddots & \vdots & \vdots \\
a_lc_{m-1} & + & b_md_{l-1} & = 1 & \text{Coefficient of } x^0
\end{array}$$

We can solve these equations by Cramer's rule to give the determinant with some coefficients in the numerator and the determinant of the Sylvester matrix in denominator to give:

$$A' = \frac{A}{\mathsf{Res}(f, g)}$$

$$B' = \frac{B}{\mathsf{Res}(f, g)}$$

for some polynomials A and B, which we can find as we know A', B' from Extended Euclidean Algorithm. Now from $A'f + B'g = 1$ it follows that for such A, B, we have $Af + Bg = \mathsf{Res}(f, g)$ where $\deg(A) = \deg(A')$, $\deg(B) = \deg(B')$. □

Definition 2.9 (Discriminant)**.** *We define the discriminant of a polynomial $f(x) = a_0 + a_1x + \ldots a_lx^l$ as*

$$disc(f) = \frac{(-1)^{\frac{l(l-1)}{2}}}{a_l}\mathsf{Res}(f, f')$$

where f' is the derivative of $f(x)$ wrt x.

Notice that discriminant of a polynomial $f(x)$ being zero would imply that it is square full since f and f' have a common factor. We can extend this notion to multivariates as well.

Given two polynomials $f(x_1, x_2)$ and $g(x_1, x_2)$, we can compute $\mathsf{Res}_{x_2}(f, g)$, the resultant w.r.t. x_2, in a similar fashion by considering x_1 as a constant. The resulting polynomial will be a polynomial in x_1, and the roots r of that polynomial would give the values of x_1 for which $f(r, x_2), g(r, x_2)$ have common factors.

Let $f'(x_1, x_2)$ be some first-order derivative of $f(x_1, x_2)$. Let the resultant of f and f' w.r.t. x_2 be $R(x_1) := \mathsf{Res}_{x_2}(f(x_1, x_2), f'(x_1, x_2))$, which is also one of the discriminant of f. $R(x_1)$ being identically zero would imply: f and its derivative have a common factor.

More literature on resultants and elimination theory can be found in [CLO13, Chap.3].

2.2.2 Basic algebraic geometry

We are now going to describe some algebraic geometry terminologies and definitions that will be used later in the thesis. For a field k, we define *affine space* as follows:

$$\mathbb{A}_k^n = \{(c_1, c_2, \dots c_n) | c_i \in k\} \tag{2.2}$$

Now if $S \subset k[x_1, x_2, \dots x_n]$ be a collection of polynomials, the ideal generated by elements of S is called $\mathtt{I}_S$. We define the *affine variety* as:

$$\mathbf{V}(S) = \{(v_1, v_2, \dots v_n) \in \mathbb{A}_k^n | p(v_1, v_2, \dots v_n) = 0 \ \forall p \in S\} \tag{2.3}$$

It directly follows that $\mathbf{V}(S) = \mathbf{V}(\mathtt{I}_S)$.

We also define an ideal $\mathtt{I}$ over a zero set $Z \subseteq k^n$ as

$$\mathtt{I}(Z) = \{f \in k[x_1, x_2, \dots x_n] | f(a) = 0 \ \forall a \in Z\} \tag{2.4}$$

We also define *radical* of an ideal I, denoted as $\sqrt{I}$ given by:

$$\sqrt{I} = \{f \in k[x_1, x_2, \dots x_n] | \exists m \in \mathbb{N}; f^m \in I\} \tag{2.5}$$

It can be shown that radical of an ideal is also an ideal.

Now, for any two ideals $a, b \in k[x_1, x_2, \dots x_n]$, *Zariski Topology* states that $a \subseteq b \implies V(a) \supseteq V(b)$.

Lemma 2.10. *The following relations hold true:*

1. $V(\phi) = k^n$, $V(k[x_1, \dots x_n]) = \phi$
2. $V(ab) = V(a \cap b) = V(a) \cup V(b)$
3. $V(\sum_{i \in I} a_i) = \bigcap_{i \in I} V(a_i)$ *for a family of ideals* $(a_i)_{i \in I}$

Proofs of these statements can be found in [**Mil17**].

Another important problem in mathematics is Hilbert's Nullstellensatz (HN), translated as 'the theorem of zeroes', which establishes a connection between the existence of zeroes of a system of polynomials with the ideal formed by those polynomials over an algebraically closed field.

Theorem 2.11 (Weak HN)**.** *For an ideal* $I \subseteq \mathbb{K}[x_1, x_2, \dots x_n]$, $V(I) = \phi \iff 1 \in I$.

Theorem 2.12 (Strong HN)**.** *For every ideal* $I \in \mathbb{K}[x_1, x_2, \dots x_n]$, $\sqrt{I} = I(V(I))$.

It can be shown that Weak HN implies Strong HN and vice versa. More explorations into Hilbert's Nullstellensatz can be found in [Dwi17].

2.2.3 Commutative algebra preliminaries

The following lemma gives an estimate on the number of roots of an *absolutely* irreducible polynomial, which has been used in [HW99] to find a root of a system of polynomial equations over $\mathbb{F}_q$. This is the reason why absolute irreducibility is crucial in this paper.

Theorem 2.13 (Number of roots [Sch74])**.** *An absolutely irreducible polynomial* $f(\mathbf{x})$ *(d-degree n-variate) has number of roots in the range,* $q^{n-1} \pm ((d-1)(d-2)q^{n-1.5} + 6d^2q^{n-2})$*, over a 'large' finite field* $\mathbb{F}_q$ *(namely,* $q > \omega(n^3d^5)$*).*

Gröbner basis. We require some concepts of Gröbner basis in our algorithm to find 'special' lifts to $\widehat{\mathbb{G}}$ (in Lemma 5.3). Modulo multivariate polynomial ideals, the remainder on division is not always unique. Thus, we modify the ideal by adding some more generators, depending on a given *ordering* of variables, such that the remainder modulo the ideal is unique.

For a given ideal $\mathtt{I}$, the *S-polynomial* of two polynomials g_1, g_2 in $\mathtt{I}$ is defined as

$$S(g_1, g_2) = \frac{\mathrm{lcm}(\mathrm{LM}(g_1), \mathrm{LM}(g_2))}{\mathrm{LT}(g_1)} \cdot g_1 \; - \; \frac{\mathrm{lcm}(\mathrm{LM}(g_1), \mathrm{LM}(g_2))}{\mathrm{LT}(g_2)} \cdot g_2 \; , \tag{2.6}$$

where LM denotes the leading monomial and LT denotes the leading term.

Buchberger [Buc65] gave the famous algorithm to compute the (reduced) Gröbner basis; by considering every pair of current generators of the ideal and iteratively adding their S-polynomials; until the S-polynomials are zero. More properties of Gröbner basis, and their complexity, can be found in [CLO13].

2.3 Factorization modulo prime powers

Hensel's lifting was given by Kurt Hensel [Hen18] to 'lift' a factorization given modulo a prime ideal to modulo higher powers of that ideal. In our case we will only deal with the ideal being $\langle p \rangle_{\mathbb{Z}}$ and lifting to modulo p^k for integers $k \geq 1$. We will describe a few properties related to polynomials in rings of the form $\mathbb{Z}/p^k\mathbb{Z}$. For further reading, we refer the reader to [BS96].

Lemma 2.14. *A polynomial* $f(x) \in \mathbb{Z}[x]$ *can be uniquely written in the form* $f(x) = a(x) + p \cdot g(x)$ *for a prime* p*, where* $a(x) \in \mathbb{Z}/p\mathbb{Z}$*.*

Proof. The proof directly follows from the fact that $f(x) = \sum_i c_i x^i = \sum_i ((c_i \bmod$

$p)x^i) + \sum_i (c_i - (c_i \bmod p))x^i$ and $\mathbb{Z}/p\mathbb{Z}$ being an integral domain, implying uniqueness. □

Lemma 2.15. *A polynomial $f(x) \in \mathbb{Z}/p^k\mathbb{Z}[x]$ is an unit if and only if $f(x)$ can be written as $f(x) = a + p \cdot g(x)$ where $a \in \mathbb{F}_p^\times$.*

Proof. If $f(x)$ is an unit, it can be written as $f(x) = a(x) + p \cdot g(x)$ by Lemma 2.14. Now since this is an unit, there must be an inverse $f(x)^{-1}$ written in the form $a'(x) + pg'(x)$, product of which is one. By taking the product modulo p, we get that $a(x)a'(x) = 1$. Now since both a, a' are polynomials, there degree can not decrease after multiplication, and hence must be constants. So we get $a(x) \in \mathbb{F}_p^\times$.
For the converse, we find an inverse of $f(x) = a + p \cdot g(x)$ for $a \in \mathbb{F}_p^\times$. Applying binomial theorem, we write $(a + p \cdot g(x))^{-1} = a^{-1}(1 + p(a^{-1}g(x)))$. Now using the expansion $(1 + x)^{-1} = 1 - x + x^2 - \ldots$, we get $f(x)^{-1} = a^{-1}(1 - p(a^{-1}g(x)) + p^2(a^{-1}g(x))^2 - \cdots + (-1)^{k-1}(a^{-1}g(x))^{k-1})$. It can be checked that $f(x)^{-1}f(x)$ is indeed 1 over $\mathbb{Z}/p^k\mathbb{Z}[x]$. □

Based on these we present Hensel's lemma, which gives us a technique to lift factorizations of certain kinds of polynomials from modulo p to modulo p^k.

Theorem 2.16. *Let $f, g, h \in \mathbb{Z}[x]$ be polynomials such that $f(x) \equiv g(x)h(x) \bmod p$, and $gcd(g(x) \bmod p, h(x) \bmod p) = 1$ in $\mathbb{Z}/p\mathbb{Z}[x]$, then there exists polynomials (referred to as 'lifts') $\tilde{g}, \tilde{h} \in \mathbb{Z}[x]$ such that $f(x) \equiv \tilde{g}(x)\tilde{h}(x) \bmod p^k$ $\forall k \in \mathbb{N}$, and $\tilde{g} \equiv g \bmod p$, $\tilde{h} \equiv h \bmod p$.*

Proof. We give an algorithm to prove this, which has also been explained in [BS96]. Since gcd of g, h over $\mathbb{F}_p$ is one, $\exists \lambda, \mu \in \mathbb{F}_p[x]$ such that $\lambda g + \mu h \equiv 1 \bmod p$. From this we iteratively construct a factorization of f modulo higher powers of p as follows.
The proof of correctness is by induction on i. At every step, each of the factors are congruent to the previous factor modulo p. Suppose after update at i^{th} step, g, h were g_{i-1}, h_{i-1}, and become updated to $\tilde{g}, \tilde{h}$ respectively. Then we have $f - \tilde{g}\tilde{g} \equiv f - (g_{i-1} + p^{i-1}u)(h_{i-1} + p^{i-1}v) \bmod p^i$. From this, if we substitute the values of u, v and consider the fact that g_{i-1}, h_{i-1} are coprime modulo p (since $\lambda g + \mu h \equiv 1 \bmod p$),

Algorithm 1 Hensel's Lifting

1: **for** $i \in 2, 3, \ldots k$ **do**
2: $\quad q := \frac{f-gh}{p^{i-1}} \bmod p$
3: $\quad u := q\mu$
4: $\quad v := q\lambda$
5: $\quad g := g + p^{i-1}u$
6: $\quad h := h + p^{i-1}v$
7: **return** $\tilde{g} = g, \tilde{h} = h$

we can show that this expression is zero modulo p^i. It can also be shown that $\tilde{g}, \tilde{h}$ are coprime over p, and the corresponding λ and μ can be found. For the complete proof, we refer the reader to [BS96]. □

Corollary 2.17. *Hensel's lifting is unique upto multiplication by units.*

However note that Hensel's lifting can not proceed if g, h has some non-trivial gcd modulo p. This basically means that $f(x) \bmod p$ is a perfect power of some irreducible. [GH96] gives an analysis of the difficulties we face in this case. We now show a method from [GH98] which shows some of the conditions that need to be satisfied in order to lift.

Theorem 2.18 ([GH98])**.** *Let $f \equiv gh \bmod p^k$ such that $f \equiv \phi^\ell \bmod p$, where ϕ is an irreducible polynomial modulo p, and $e \leq \ell/2$ such that $g \equiv \phi^e \bmod p$, $h \equiv \phi^{\ell-e} \bmod p$. Then the following are equivalent:*

1. *$\frac{f-gh}{p^k} \in \mathbb{Z}[x]$ and divisible by g^e over modulo p*
2. *For every $\psi \in \mathbb{Z}[x]$ with $deg(\psi) < deg(g)$, there is a polynomial $\theta \in \mathbb{Z}[x]$ with $deg(\theta) < deg(h)$ such that $f \equiv (g + p^k\psi)(h + p^k\theta) \bmod p^{k+1}$*
3. *There exist polynomials $\psi, \theta \in \mathbb{Z}[x]$, with $deg(\psi) < deg(g)$, $deg(\theta) < deg(h)$ such that $f \equiv (g + p^k\psi)(h + p^k\theta) \bmod p^{k+1}$*
4. *There exist polynomials $\psi, \theta \in \mathbb{Z}[x]$ with $f \equiv (g + p^k\psi)(h + p^k\theta) \bmod p^{k+1}$*

Proof. $(i) \implies (ii)$ Let $\frac{f-gh}{p^k} \equiv \phi^e\alpha \bmod p$ for some $\alpha \in \mathbb{Z}[x]$, and $\psi, \theta \in \mathbb{Z}[x]$, with $deg(\psi) < deg(g)$. Let $\theta \equiv \alpha - g^{\ell-2e}\psi \bmod p$. Then, using the fact that $e \leq \ell/2$, we

can show that $f - (g + p^k\psi)(h + p^k\theta) \equiv 0 \bmod p^{k+1}$.

$(ii) \implies (iii) \implies (iv)$ is directly follows. Now, we are required to show $(iv) \implies (i)$. Let $\psi, \theta \in \mathbb{Z}[x]$ with $f \equiv (g + p^k\psi)(h + p^k\theta) \bmod p^{k+1}$. Then we have

$$\frac{f - gh}{p^k} \equiv \psi g + \theta h \equiv \psi g^{\ell-e} + \theta g^e \equiv g^2(\psi g^{\ell-2e} + \theta) \bmod p$$

This proves the theorem. □

From these we see that if the polynomial to be factored is not a power of some irreducible modulo p, then we can lift it to modulo any p^k. There will be unique factors (unique upto multiplication by units) and there is an one-one correspondence of roots modulo p with roots modulo p^k. However the more difficult case is left, when we have $f \equiv \phi^\ell \bmod p$ for some polynomial $\phi(x)$ irreducible modulo p. There have been some attempts to factorize polynomials of this form [Sir17; DMS19], the best being [DMS19], which achieved factorization up to modulo p^4.

2.4 Root finding of univariates $\bmod p^k$

Definition 2.19. *A representative of a ring R, denoted by the symbol $*$, takes all values in the ring R. Formally, it is the set $* := \{a | a \in R\}$.*

We further define the operations:

- $b + * = \{b + a | a \in *\}$ for $b \in R$,
- $b* = \{ba | a \in *\}$ for $b \in R$.

Using this definition, for $\beta + p^\ell * \subseteq \mathbb{Z}/p^k\mathbb{Z}$ for $\beta \in \mathbb{Z}/p^\ell\mathbb{Z}$, $\ell \leq k$, we have

$$\beta + p^\ell * = \{\beta + p^\ell a | a \in \mathbb{Z}/p^{k-\ell}\mathbb{Z}\} \tag{2.7}$$

In a similar fashion, a *representative root* of a polynomial $f(x) \in \mathbb{Z}/p^k\mathbb{Z}$ is denoted by a set $\beta + p^\ell *$ for $\beta \in \mathbb{Z}/p^\ell\mathbb{Z}$, $\ell \leq k$ such that for any $A \in \beta + p^\ell *$,

we have $f(A) \equiv 0 \bmod p^k$. The *length* of this representative root is the number of precision coordinates of the fixed part β, which is ℓ.

For more properties of representative roots, we direct the reader to [Pan95; BLQ13; DMS21; GCM21].

Solving univariates simultaneously. Using this compact notation, we present the standard Algorithm 2 to find *all* the roots of a univariate polynomial $f(x) \in \mathbb{Z}/p^k\mathbb{Z}$; which is due to [Pan95; BLQ13; DMS21]. However, as required in this paper, we give a slight modification where we solve a system of univariates modulo p^k. The algorithm starts with the input array $(f_1, \ldots, f_r, p, k, \ldots, k)$. This can be seen as a slight modification of the Root-Find algorithm ([DMS21, Algorithm 1]) where instead of looping only over the roots of the polynomial, we loop over the common roots in order to find a root of *all* the polynomials in the system.

Algorithm 2 Root finding of $f_1(x), \ldots, f_r(x) \bmod p^k$

procedure ROOT-FIND-BLQ$(f_1, \ldots, f_r, p, k_1, \ldots, k_r)$
  if $r = 0$ **then return** $*$
  if $\exists i$ such that $f_i(x) \equiv 0 \bmod p^{k_i}$ or $k_i = 0$ **then**
    return ROOT-FIND-BLQ$(f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_r, p, k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r)$
  $R :=$ roots of $\gcd\{f_i(x) \bmod p \mid i \in [r]\}$ [Eg. use Cantor-Zassenhaus' algorithm [CZ81]].
  if $R == \phi$ **then return** ϕ
  $S := \phi$
  for $a \in R$ **do**
    $\tilde{f}_i(x) := p^{v_i} f_i(a + px)\ \forall i \in [r]$, where $v_i = v_p(f_i(a + px))$.
    $R_a := $ ROOT-FIND-BLQ$(f_1, \ldots, f_r, p, k - v_1, \ldots, k - v_r)$
    $S := S \cup (a + pR_a)$
  return S

The correctness of Algorithm 2 directly follows from [BLQ13, Corollary 4], where they prove the correctness for a single polynomial. [BLQ13, Corollary 4] also states

that the number of representative roots is at most d many when only a single polynomial is considered. Another exploration into this algorithm can be found in [Dwi22].

Theorem 2.20. *Algorithm 2 runs in randomized* $\mathsf{poly}(\max_i \deg(f_i),\ \log p, k)$ *time and returns at most d-many representative roots.*

Chapter 3

Describing the roots of multivariates modulo prime powers

In this chapter, we prove Theorem 1.1 by giving an algorithm that returns all the roots of $f(x_1, x_2)$ modulo p^k.

Theorem 1.1 (Describing the roots)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[\mathbf{x}]$ of degree d and n variables, we can efficiently describe all the roots of $f(\mathbf{x}) \equiv 0 \bmod p^k$ in deterministic $\mathsf{poly}(k^d, p^d, d^d)$ time, where $d, \log p$ are small.*

3.1 Overview of the algorithm

We prove Theorem 1.1 by giving an algorithm that returns all the possible roots modulo p^k, of a given degree-d polynomial $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$. It designs a compact data structure for this, and outputs what we call *representatives*.

The root-finding is done iteratively: find each p-adic coordinate which is a root at each step, and perform lifting to higher coordinates; as is done in the case of univariates [BLQ13; DMS19; DMS21; NRS17]. If (a_1, a_2) is a root of $f(x_1, x_2) \bmod p$, then we transform the polynomial to another polynomial given by $f(a_1 + px_1, a_2 + px_2)$. In order to find $\mathbb{F}_p$-roots of this polynomial in the next step, we remove

the 'extra' p-powers by dividing by p^v; where $v := v_p(f(a_1 + px_1, a_2 + px_2))$ is the 'val-multiplicity' and $v_p(\cdot)$ is the p-valuation (Definition 2.2). We define this step, of transforming the coordinates and subsequent division by the p-power, as the *lifting step* or *lifting of roots*. The polynomial will be modified at each step such that its $\mathbb{F}_p$-root returns a coordinate of the final (p-adic resp. $\mathbb{Z}/p^k\mathbb{Z}$) root. Notice that if (a_1, a_2) is an $\mathbb{F}_p$-root of $f(x_1, x_2)$, and after lifting, the polynomial becomes $\tilde{f}(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$ which has an $\mathbb{F}_p$-root (b_1, b_2), then $(a_1 + pb_1, a_2 + pb_2)$ is a root of $f(x_1, x_2)$ mod p^{v+1}. The *uni*variate case of this lifting technique, as developed in [BLQ13; DMS21], is explained in Section 2.4.

However, it might so happen that root (a_1, a_2) in the lifting process does *not* lift to higher powers of p; but some other root does lift, as illustrated by the following example.

Example 3.1. *Consider* $f(x_1, x_2) := x_1^3 - x_2^3 + 3x_2 - 3x_1 + 5$ *and* $p := 5$. $(1, 1)$ *and* $(2, 2)$ *are its* $\mathbb{F}_p$*-roots. Starting with the root* $(1, 1)$*, the process of lifting given by the transformation* $(x_1, x_2) \mapsto (1 + 5x_1, 1 + 5x_2)$ *and division by* 5*, yields the polynomial* $25x_1^3 - 25x_2^3 + 15x_1^2 - 15x_2^2 + 1$ *which does not have* $\mathbb{F}_5$*-roots. Although, restarting with the root as* $(2, 2)$ *yields the polynomial* $25x_1^3 - 25x_2^3 + 30x_1^2 - 30x_2^2 + 9x_1 - 9x_2 + 1$ *after lifting.* $(1, 0)$ *is now its* $\mathbb{F}_5$*-root! This anomaly is explained by a curious fact:* $(1, 1)$ *is a singular root while* $(2, 2)$ *is non-singular.*

Thus, we iteratively loop over all the possible roots at each step, by fixing one variable, say x_1, with p-many possibilities, and finding the possible d-many (or p-many) values of x_2.

3.1.1 Val-multiplicity vs valuation

For a polynomial $f(\mathbf{x})$, we define the *effective polynomial* as $f(\mathbf{x})$ mod p, where the coefficients are in $\mathbb{F}_p$ (w.l.o.g. $f(\mathbf{x})$ mod p is non-constant). Similarly, the *effective degree* of $f(\mathbf{x})$ is the degree of $f(\mathbf{x})$ mod p. Unless specified otherwise, we will denote $d_1 \geq 1$ as the effective degree of the polynomial at that step of lifting, while $d \geq d_1$ will be the total degree. ($d_1 = 0$ is trivial to handle.)

We define a *local root* of $f(\mathbf{x})$ as a root of the effective polynomial $f(\mathbf{x})$ mod p. For a local root $\mathbf{a} \in \mathbb{F}_p^2$, *local valuation* is defined as $v_p(f(\mathbf{a}))$. Similarly, *val-multiplicity of local root* is defined as $v_p(f(\mathbf{a}+p\mathbf{x}))$, i.e., the minimum valuation of the coefficients of the polynomial thus formed; we sometimes shorten it to val-mult($\mathbf{a}$). Obviously, val-multiplicity is at most the local valuation.

3.1.2 Branching w.r.t. val-multiplicities

As we have seen in Example 3.1, different $\mathbb{F}_p$ roots in steps of lifting can give rise to different val-multiplicities. Thus, we create a *tree* (Fig.3.1), having nodes as polynomials, say $f_j(\mathbf{x})$ in the j-th step of lifting, and branches arising from it corresponding to each local root $\mathbf{a} \in \mathbb{F}_p^2$. The children of this node will be the polynomials obtained from lifting by the root corresponding to the branch, given by $f_{j+1}(\mathbf{x}) := p^{-v} f_j(\mathbf{a} + p\mathbf{x})$; where $v :=$ val-mult($\mathbf{a}$).

Theorem 3.2-(1) shows that val-multiplicity is at most the effective degree (denoted d_1). Since each local root corresponds to some val-multiplicity in $[d_1]$, we associate these roots with their val-multiplicities. The branches of roots with the same val-multiplicity v, yield similar properties on the structure of the polynomial after lifting. So, we denote them as a single 'thick' val-multiplicity v branch (but they are in fact computed in several parallel val-multiplicity v branches and their corresponding polynomials as nodes). Note: There are at most p^2 local roots in $\mathbb{F}_p^2$.

The recursive steps of finding each precision coordinate can thus be seen as a tree, with each branch/child seen as a root $\mathbf{a} \in \mathbb{F}_p^2$ of val-multiplicity in $[d_1]$. If $f_j(x_1, x_2)$ mod p is a d_1-form at $\mathbf{a}$, then it is in the ideal $\langle x_1 - a_1, x_2 - a_2 \rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]}$ (Lemma 3.7). This is our algorithm's hard case of a root with the maximum val-multiplicity, namely d_1; so we need to branch into a special val-multiplicity $= d_1$ branch/node in the tree. We 'stay' here till all the chains of val-mult$=d_1$ roots are explored and stored in an array (D in Algo.3). This is done in the red subpart of the tree in Fig.3.1. Next, it branches into the easier degree-reduction cases; which is denoted by the green nodes (enclosed by the left rectangle in Fig.3.1).

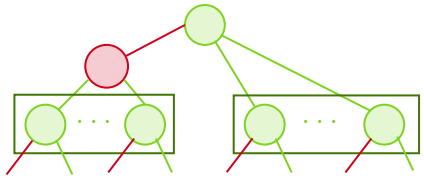


Figure 3.1: Branching along the tree

As we go down the tree, we perform lifting computationally; in a way, depth h implies that the polynomial has been lifted h times. This has been schematically portrayed in Figure 3.1, where the red node is a more complicated transformation than the simple lifting (done in green node).

To summarize, the *degree reduction case* of local root is where effective degree reduces ($v < d_1$ suffices, due to Theorem 3.2); while *val-multiplicity* d_1 *case* is that when val-mult $v = d_1$.

3.1.3 Degree reduction

The crux of Algorithm 3 lies in our Theorem 3.2 which states that optimistically the effective degree reduces "most of the time". The algorithm ends when we either have exhausted the power of p, denoted by k, or when the effective degree becomes 1. The latter case has been explained in more details in Theorem 3.2, and the result implies that a root always exists once the effective degree has fallen down to 1; which is essentially (p-adic) Hensel's lifting on a *linear* polynomial [Hen18].

Since we have degree reduction, we expect the number of steps (or the tree-depth) to be $O(d_1)$. However, the difficulty arises when the effective degree remains unchanged. Here, in the worst-case, factors of p^{d_1} are divided out of the polynomial in lifting steps, and the depth of the branching tree (Figure 3.1) can extend to $\Omega(k/d_1)$, which we can not afford as it would lead to iteratively going over $p^{\Omega(k/d_1)}$-many local roots; which is like the complexity of the brute-force algorithm.

3.1.4 Hensel's lifting

Given a non-singular root $\mathbf{a}$ of polynomial $h(\mathbf{x})$, we can lift it to modulo any p-power (like in Theorem 3.2-(2)), using a variant of p-adic Hensel's lifting [Hen18]. Since

a is a non-singular root, at least one of the first-order derivatives of $h(\mathbf{x})$ will not vanish. Corollary 3.5 implies that the val-multiplicity is then exactly 1, and in the next step of lifting, the effective polynomial will be *linear*. If this linear polynomial is of the form $m_1x_1 + m_2x_2 + m_0$, then (say) we can fix x_2 to any value in $[p]$ and find the corresponding unique value of x_1 to yield a root by simple lifting. For the next p-adic coordinate, after lifting, these m_1, m_2 (coefficients of x_1 resp. x_2) will not change; while m_0 might change (due to the lifting performed in the non-effective part of the polynomial). Thus, from Theorem 3.2-(2), we have the fact that the effective polynomial continues to stay as linear, and we can fix the current-coordinate x_2 to find the corresponding x_1 every time; enabling us to lift to modulo any p-power (for arbitrary fixing of x_2 in this example).

3.1.5 Avoiding $\Omega(k)$ lifting steps

In Theorem 3.2, we show that the effective degree does not reduce *only if* the val-multiplicity of the local root is d_1. So, we consider the branches of degree reduction as discussed in previous paragraphs, but tweak our algorithm carefully in val-mult $= d_1$ case to prevent the depth of the tree (and the number of lifting steps) from blowing-up to $\Omega(k/d_1)$.

We want branches of lesser val-multiplicity ($< d_1$) originating from the branches of val-multiplicity $= d_1$ roots, where the effective degree reduces in each of them. So, we create a process of 'removing' these contiguous val-multiplicity d_1 nodes, instead of looping over all of them; and recursively call the root-finding function on each of the degree-reducing branches arising from this single node/polynomial. This removal-process is guided by something called d_1-*forms* (Lemma 3.7), and will be subdivided into single and multiple val-multiplicity d_1 roots. (1) When *multiple* val-multiplicity roots exist, we show in Lemma 3.8 that the polynomial has a special form (namely d_1-*power*). We traverse these cases in a contiguous way. (2) Next we traverse over cases where val-multiplicity d_1 root is unique. At the end, when we encounter lesser val-multiplicity roots, we recursively call the root-finding algorithm.

Overall, we find the lengths of these contiguous traversals, as well as the possibilities of the underlying d_1-powers resp. unique-roots. This is discussed in the latter-half of Section 3.3, by considering a dynamic basis-change on the variable set $\mathbf{x}$, so that we do not have to iterate over 'too many' local roots. This is done by employing the idea of [BLQ13] to find representative-roots of a univariate polynomial system.

Summarizing this case of val-mult=d_1 roots, we showed that the possibilities of contiguous chains is small (i.e. polynomial in k, d), and every lesser val-multiplicity branch appearing from these chains is in a degree reduction case. So, in the tree (Fig.3.1) an intermediate red node is created that 'jumps' over all the val-mult=d_1 cases (Sections 3.3–3.4). This bounds the depth of our tree to $2d$.

3.1.6 Stopping condition and representative roots

The algorithm terminates when either a root gets completely specified $\bmod p^k$, or when effective degree ≤ 1 (any of its roots can be Hensel lifted all the way to our required power of p), or when no root exists. In the third case, the root-set returned is just the emptyset ϕ, while in the first case it is a singleton.

For the second case, roots will be returned in terms of representative roots (Definition 2.19). Eg. when the lifted polynomial is zero modulo p^ℓ, any value in $\mathbb{Z}/p^\ell\mathbb{Z}$ is a root, and thus we return $*_1$ resp. $*_2$ for the coordinates x_1 resp. x_2, which represent the entire $\mathbb{Z}/p^\ell\mathbb{Z}$. The roots returned will be $(*_1, *_2)$, with the number of possibilities being $p^{2\ell}$. This will be termed as our usual *representative root*.

When the effective degree is 1: as we sketched before, we can fix one variable as a local root and find the value of the other variable. In such a fashion, given any value of one coordinate, say x_1, we can find each p-adic coordinate of x_2 one by one. Even if only one variable is present in the linear form, say x_2, the other variable x_1 will still be free, and for any given value of x_1, denoted by $*$, we can find the corresponding values of the local roots of x_2, and thus a root of the polynomial modulo p^ℓ. Let us denote this function for determining x_2 from any value of x_1 by $c(\cdot)$, which simply finds each coordinate of x_2 using Hensel's lifting. Thus, the

output can be denoted as $(*, c(*))$. The number of roots in this expression is p^ℓ. (Note: This can contribute more roots to the original f mod p^k, so a more careful calculation is done in Section 4.2.) This type of representative root will be termed as *linear-representative*.

3.1.7 Main algorithm

Based on these ideas, we sketch our Algorithm. Its main procedure is the ROOT-FIND() function in Algorithm 3. It takes as *input*: the polynomial $f_j(x_1, x_2)$ and the number p^{k_j} ($k =: k_0$ initially). The algorithm starts with calling ROOT-FIND$(f(x_1, x_2), p^k)$. If there are valid roots, it outputs the set of roots $R \subseteq (\mathbb{Z}/p^k\mathbb{Z})^2$, otherwise returns ϕ.

The submodule of CREATE-WALK() in Algorithm 4 is a procedure to eliminate intermediate computations where effective degree does not decrease. In a way, it speeds-up the search for roots to higher precision coordinates, by jumping over contiguous cases of roots of val-multiplicity d_1. CREATE-WALK() outputs an *array* of: linear transformation which can be used to jump over the val-multiplicity d_1 roots, or linear-representative root which directly becomes part of the output.

Algorithm 3 Root Finding of $f_j(x_1, x_2)$ mod p^{k_j}

1: **procedure** ROOT-FIND$(f_j(x_1, x_2), p^{k_j})$
2: **if** $k_j \leq 0$ OR $f_j(x_1, x_2) \equiv 0 \bmod p^{k_j}$ **then return** $(*_1, *_2)$
3: Define $d_1 := \deg(f_j \bmod p)$, $R := \phi$.
4: **if** $d_1 = 1$ **then**
5: **return** linear-representative $(*, c(*))$ or $(c(*), *)$, where $c(\cdot)$ is given by Hensel's Lifting.
6: **for** $a_1 \in \{0, p-1\}$ **do**
7: **for** a_2 such that $f_j(a_1, a_2) \equiv 0 \bmod p$ and val-mult(**a**)$< d_1$ **do**
8: $f_{j+1}(x_1, x_2) := p^{-v} f_j(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f_j(a_1 + px_1, a_2 + px_2))$.
9: $S :=$ ROOT-FIND$(f_{j+1}, p^{k_j - v})$

10: $R := R \cup (\mathbf{a} + pS)$

11: **if** val-multiplicity= d_1 root exists **then**

12: $D :=$ CREATE-WALK(f_j, p^{k_j})

13: **for** $(r_1 + p^{i_1}L_1, r_2 + p^{i_2}L_2, i_3) \in D$ **do**

14: Write f_j in basis $\{L_1, L_2\}$ to get $\tilde{f}_j(L_1, L_2) := f_j(x_1, x_2)$.

15: Lift it to $\tilde{f}_j(L_1, L_2) := p^{-i_3 d_1} \cdot \tilde{f}_j(r_1 + p^{i_1}L_1, r_2 + p^{i_2}L_2)$.

16: **if** $k_j - i_3 d_1 \leq 0$ **then**

17: The roots will be $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ in (L_1, L_2) basis.

18: Consider the tuple $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ and perform the inverse linear transformation from (L_1, L_2) to (x_1, x_2) on this tuple as a whole. Store this representative root (with two independent $*$'s) in a set S

19: $R := R \cup S$

20: **else**

21: For $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$, find the val-mult $< d_1$ local roots and then recursively find all the roots; as done in Steps 6-10. Let this be given by the set $\tilde{R}$.

22: For each root $(\tilde{r}_1, \tilde{r}_2) \in \tilde{R}$ of $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$: consider $(r_1 + p^{i_1}\tilde{r}_1, r_2 + p^{i_2}\tilde{r}_2)$ and perform inverse linear transformation from (L_1, L_2) to (x_1, x_2) on them. Store these final roots $(\bmod p^{k_j})$ in a set S.

23: $R := R \cup S$

24: **return** R

3.2 Degree reduction: Polynomial after lifting

In this section, we analyze the effective degree at each step and look more closely as to when this decreases, or remains the same, by looking at the val-multiplicity of the local root during lifting. The proof idea is to analyze the monomials in

terms of x_1 and x_2, and see how they behave after the transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$ followed by division by appropriate power of p. This can be summed up by the following theorem.

Theorem 3.2 (Degree reduction)**.** *For a polynomial $f(x_1, x_2) \in (\mathbb{Z}/p^k\mathbb{Z})[x_1, x_2]$, given an $\mathbb{F}_p^2$-root (a_1, a_2) of $f(x_1, x_2)$, let us denote $g(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f(a_1 + px_1, a_2 + px_2))$. Let the previous effective degree be $d_1 := \deg(f(x_1, x_2) \bmod p)$ and current effective degree be $d_2 := \deg(g(x_1, x_2) \bmod p)$. Then the following holds:*

1. *If $d_1 > 1$, then $d_2 \leq v \leq d_1$. (So, $d_2 = d_1$ only if $v = d_1$.)*
2. *If $d_1 = 1$, then $d_2 = 1$.*

Before proving this, let us first see the degree evolution in some concrete examples.

Example 3.3. *Let us see how the effective degree could* reduce*. Consider $f(x_1, x_2) = x_1^2 + x_2^3 \bmod p$. This has degree $d_1 = 3$. Clearly, $(0, 0)$ is its root modulo p. So, apply the transformation $(x_1, x_2) \mapsto (0 + px_1, 0 + px_2)$, to get $g(x_1, x_2) := p^{-2} f(px_1, px_2) = x_1^2 + px_2^3$, which has effective degree $d_2 = 2 = v < d_1$.*

Example 3.4. *Let us see why the effective degree might remain* unchanged *in 'many' steps (which is bad for us). Consider $f(x_1, x_2) = x_1^3 + x_2^3 + p^9(x_1 + x_2 - 1) \bmod p^{10}$. Then, around its root $(0, 0)$, use the translation $(x_1, x_2) \mapsto (0 + px_1, 0 + px_2)$ and division by p^3, to reduce to a simpler $g(x_1, x_2) := x_1^3 + x_2^3 + p^6(-1) \bmod p^7$. We do this two more times to reduce to a simpler $g(x_1, x_2) := x_1^3 + x_2^3 - 1 \bmod p$. In these three steps, the degree $= 3 = v$ did not reduce. However, if $p \neq 3$, then the degree will finally reduce in the fourth step. [If $p = 3$ then proceed with $g := x_1 + x_2 - 1$.]*

Proof of Theorem 3.2. We have two cases.

Case 1: $\mathbf{d_1 > 1}$. We have a local root (a_1, a_2) such that $v = v_p(f(a_1 + px_1, a_2 + px_2))$. Using Taylor's expansion (Definition 2.1), we can write the polynomial in the

form (say over $\mathbb{Z}_p$)

$$f(a_1 + px_1, a_2 + px_2) \;=\; \sum_{\ell=0}^{d} \left(\sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!} \cdot (px_1)^{i_1} (px_2)^{i_2} \right), \tag{3.1}$$

where $d := \deg(f)$. The terms in Equation 3.1 need to vanish modulo p^v for all $\ell \leq v$. In particular, $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$. Suppose $v > d_1$, then by the above equation $f(a_1 + x_1, a_2 + x_2) \equiv 0 \bmod p$, implying $f(\mathbf{x}) \equiv 0 \bmod p$, which contradicts with $d_1 > 1$. Thus, $v \leq d_1$.

However, we do have a term which has valuation exactly v (= val-multiplicity of the local root), and this can be obtained only from monomials where $i_1 + i_2 \leq v$ (that too in the effective polynomial part). So, the highest degree term surviving among these (in $g \bmod p$) has degree $d_2 \leq v \leq d_1$.

Remark: The case of $d_2 = d_1$ implies that $v = d_1$. Thus, $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$ for all orders $|\mathbf{i}| < d_1$; while, some order-d_1 partial-derivative at $\mathbf{a}$ has p-valuation exactly 0.

Case 2: $\mathbf{d_1 = 1}$ (Hensel's Lifting)**.** Write $f(x_1, x_2) =: f_1(x_1, x_2) + p \cdot f_2(x_1, x_2)$. We have the effective polynomial $\deg(f_1(\mathbf{x})) = 1$, and hence it can be written as a linear polynomial $m_1 x_1 + m_2 x_2 + m_0$. Since (a_1, a_2) is a local root, we transform f to get $m_1(a_1 + px_1) + m_2(a_2 + px_2) + m_0 + p \cdot f_2(a_1 + px_1, a_2 + px_2)$. Dividing by p, and going $\bmod p$, we get in the next step to another linear polynomial: $m_1 x_1 + m_2 x_2 + m_0'$. So we end up with $d_2 = 1$. □

Using the proof of Theorem 3.2, we get a corollary on partial derivatives of $f(\mathbf{x})$, which motivates the inclusion of the term 'multiplicity' in our new concept of 'val-multiplicity'.

Corollary 3.5. *Local root* $\mathbf{a}$ *of* $f(\mathbf{x})$ *has val-multiplicity* $\geq v$*, if and only if* $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$*, for all orders* $|\mathbf{i}| < v$*.*

Remark 3.6. *Implicitly, the above proofs of Theorem 3.2 and Corollary 3.5 needed* $d < p$ *so that the factorials could appear in the denominators (in Equation 3.1).*

For smaller p, the same proof works 'syntactically'. Formally, consider the Hasse derivatives *instead of partial-derivatives.*

Using this theorem, we can get an idea of how the effective degree reduces. If a root $(a_1, a_2) \in \mathbb{F}_p^2$ is such that $f(a_1 + px_1, a_2 + px_2) \not\equiv 0 \bmod p^{d_1}$, then we can keep applying the appropriate transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$, until the effective degree reduces to 1. Once this effective degree has reduced to 1, we have a *compact description* of all its roots: as we can arbitrarily fix one variable and uniquely find the p-adic value of the other variable.

However, the problem arises when the root (a_1, a_2) is such that $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$. In this case, the degree may not reduce, and we might need to keep lifting to k/d_1 steps. This is computationally infeasible, the search-tree becomes very large, and takes time exponential in k/d_1. We tackle this case in the next section.

3.3 Structure of polynomial via val-mult$= d_1$ roots

We need to handle the challenge of our local root $\mathbf{a}$ of f having val-multiplicity $v = d_1$. Here, the effective degree does not reduce in the next step. We first show the structure of such $f(x_1, x_2)$.

Lemma 3.7 (d_1-form at $\mathbf{a}$)**.** *If $\mathbf{a} \in \mathbb{F}_p^2$ is a root of $f(\mathbf{x}) \bmod p$ such that $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$, where d_1 is the effective degree of f, then $f(\mathbf{x}) \in \langle x_1 - a_1, x_2 - a_2 \rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]}$.*

Proof. Recall Taylor's expansion (Definition 2.1) and Corollary 3.5. Write $f(\mathbf{x})$ as $f((x_1 - a_1) + a_1, (x_2 - a_2) + a_2) = \sum_{\ell=0}^{\infty} A_\ell$, where,

$$A_\ell := \sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!} \cdot (x_1 - a_1)^{i_1} (x_2 - a_2)^{i_2} .$$

By Corollary 3.5, we know that the A_t's, for $t < d_1$, vanish modulo p as the root has val-multiplicity $v = d_1$. Furthermore, for $t > d_1$, A_t's vanish modulo p; as f

has effective degree d_1 and this A_t has derivatives of order $> d_1$. Thus, all A_t's, apart from A_{d_1}, vanish modulo p. So, the polynomial f is of the form $\sum_i c_i \cdot (x_1 - a_1)^{d_1-i}(x_2 - a_2)^i$, which is the required d_1-*form in* $\mathbf{x} - \mathbf{a}$. □

In Lemma 3.7's situation, if $\mathbf{a}$ is unique, then using the structure of f we can easily find the root (eg. a simple search in $\mathbb{F}_p^2$), and lift without getting into multiple val-mult=d_1 branching. A serious obstruction arises when there are several local roots $\mathbf{a}$ of val-multiplicity $= d_1$. We will now show the extra special structure of such an $f(x_1, x_2)$.

W.l.o.g let $\mathbf{0}$ be a local root of val-multiplicity $= d_1$. This means that $f \in \langle x_1, x_2\rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]}$. If another local root $\mathbf{a} \neq \mathbf{0}$ exists with val-multiplicity $= d_1$, then we also have $f(\mathbf{x}) \in \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]}$. So, $f \in \langle x_1, x_2\rangle^{d_1} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}$, over $\mathbb{F}_p[\mathbf{x}]$. Then, we show f to be a *perfect-power*!

Lemma 3.8 (Two val-mult=d_1 roots)**.** *For a polynomial $f \in \mathbb{F}_p[x_1, x_2]$ of degree d_1, if $f \in \langle x_1, x_2\rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}_{\mathbb{F}_p[\mathbf{x}]}$, for some $\mathbf{a} \neq \mathbf{0} \in \mathbb{F}_p^2$, then we have $f \equiv c(a_2x_1 - a_1x_2)^{d_1} \bmod p$, where $c \in \mathbb{F}_p^*$.*

Proof. W.l.o.g., assume that $a_1 \in \mathbb{F}_p^*$. Thus, we have

$$\begin{aligned}\langle x_1 - a_1, x_2 - a_2\rangle^{d_1} &= \langle x_1 - a_1,\ a_1x_2 - a_1a_2\rangle^{d_1} \\ &= \langle x_1 - a_1,\ a_1x_2 - a_1a_2 - a_2(x_1 - a_1)\rangle^{d_1} \\ &= \langle x_1 - a_1,\ a_1x_2 - a_2x_1\rangle^{d_1}.\end{aligned} \tag{3.2}$$

Also, $\langle x_1, x_2\rangle^{d_1} = \langle x_1, a_1x_2 - a_2x_1\rangle^{d_1}$ (as $a_1 \neq 0$). The intersection of these two ideals modulo the ideal $\langle a_1x_2 - a_2x_1\rangle$ is: $\langle a_1x_2 - a_2x_1\rangle + \langle x_1(x_1 - a_1)\rangle^{d_1}$ (as x_1 and $x_1 - a_1$ are coprime mod $a_1x_2 - a_2x_1$). Since f has effective degree less than $2d_1$, we deduce: $(a_2x_1 - a_1x_2) \mid f$.

The quotient $f/(a_2x_1 - a_1x_2) \in \langle x_1, a_2x_1 - a_1x_2\rangle^{d_1-1} \cap \langle x_1 - a_1, a_2x_1 - a_1x_2\rangle^{d_1-1}$. Clearly, degree of this quotient polynomial is $d_1 - 1$. So, we can repeat this process to show that $(a_2x_1 - a_1x_2)^{d_1} \mid f$; which makes the two equal up to a constant multiple. □

Hence, we see that if a polynomial f has two val-mult=d_1 roots with one of them being zero and the other being $\mathbf{a} \neq \mathbf{0}$, then the effective polynomial $f \bmod p$ is of the form $(a_2x_1 - a_1x_2)^{d_1}$. This means that f is d_1-th power of a linear polynomial iff rank of the val-mult=d_1 roots is two (i.e. multiple such roots). In the case of unique val-mult=d_1 root we will call the polynomial d_1-*nonpower-form*, while that for multiple val-mult=d_1 roots, we call the polynomial d_1-*power*.

Branching in d_1-nonpower-form. In this case, find the unique val-multiplicity d_1 root, and do the lifting step. There is no branching required.

Branching in d_1-power. W.l.o.g. the effective polynomial will be of the form $(a_2x_1 - a_1x_2)^{d_1}$. So, there are p roots (of val-mult=d_1): (a_1t, a_2t) for any $t \in \mathbb{F}_p$. This leads to branching, which we will avoid, by taking a different route.

The first observation (Lemma 3.9) is that d_1-nonpower-form can not lead to a d_1-power. Thus, we deduce that whenever a contiguous chain of d_1-power lifting ends, then every d_1-form in the subsequent contiguous lifting steps is a d_1-nonpower-form.

Lemma 3.9 (Nonpower to power?)**.** *If f is a d_1-nonpower-form having a single val-mult=d_1 root $\mathbf{a}$, then its lift $p^{-d_1} f(\mathbf{a} + p\mathbf{x})$ is not a d_1-power.*

Proof. W.l.o.g. we can assume $\mathbf{a} = \mathbf{0}$. Since the effective polynomial is a d_1-form having $(0, 0)$ as the root, it is of the form

$$f(x_1, x_2) \equiv \sum_{i=0}^{d_1} c_i x_1^i x_2^{d_1-i} \bmod p\,. \tag{3.3}$$

After lifting given by $(x_1, x_2) \mapsto (px_1, px_2)$, followed by division by p^{d_1}, this polynomial will become

$$\sum_{i=0}^{d_1} c_i x_1^i x_2^{d_1-i} + g(x_1, x_2)\,,$$

for some polynomial g of degree $\leq d_1 - 1$. Suppose this lift is a d_1-power, say $(L+m_0)^{d_1} \bmod p$, where $m_0 \in \mathbb{F}_p$ and L is a linear form in x_1, x_2. Now comparing the degree d_1 homogeneous parts in all these equations, we conclude that $f \equiv L^{d_1} \bmod p$. This contradicts the fact that it was a d_1-nonpower-form. Therefore, d_1-nonpower-forms can not become d_1-powers in one lifting step. □

So we mainly need to study the case where: a d_1-power, say L^{d_1} is followed by another d_1-power, say L'^{d_1}, in the next lifting step. In the next subsection we unearth the structure that goes in the formation of L' after lifting the polynomial $L^{d_1} + \langle p \rangle$. This will give us the optimized bound on the branching of the red-nodes of the tree.

3.3.1 Structure of consecutive d_1-powers.

For a d_1-form, the effective polynomial $f(x_1, x_2) \bmod p$ will be of the form L^{d_1}, for some linear polynomial L (eg. $x_1 + x_2 + 1$). W.l.o.g. assume $\{L, x_2, 1\}$ to be of rank=3 (over $\mathbb{F}_p$). Let us rewrite f in the basis $\{L, x_2\}$, instead of $\{x_1, x_2\}$, denoted by $\tilde{f}(L, x_2)$ $(= f(\mathbf{x}))$. Since it is an invertible linear transformation, it now suffices to find roots of

$$\tilde{f} \;=:\; L^{d_1} + p \cdot L^{d_1-1} \cdot u_1(x_2) + p \cdot L^{d_1-2} \cdot u_2(x_2) + \cdots + p \cdot u_{d_1}(x_2)\,. \tag{3.4}$$

Lift d_1-power to d_1-power. Suppose after lifting by $p^{-d_1}\tilde{f}(pL, x_2)$, the effective polynomial is *again* a d_1-power; then it has to be the case that

$$L^{d_1} + L^{d_1-1} \cdot u_1(x_2) + L^{d_1-2} \cdot u_2(x_2)/p + \cdots + u_{d_1}(x_2)/p^{d_1-1} \;\equiv\; (L + u_1(x_2)/d_1)^{d_1} \bmod p\,, \tag{3.5}$$

for some univariate polynomials u_j's, such that Equation 3.5 is a perfect power of the linear polynomial $L + u_1(x_2)/d_1$. Consequently, those local roots a_2 for which the above system is satisfied, transform previous L to $p\,(L + u_1(a_2)/d_1)$ in this lifting step.

Considering RHS expansion, we also obtain equations, for $j \in [d_1]$, as

$$u_j(x_2) \;\equiv\; p^{j-1}\binom{d_1}{j} \cdot (u_1(x_2)/d_1)^j \bmod p^j\,. \tag{3.6}$$

Note: In case $p | d_1$, the above modulus can be further increased to clear away p-multiples from the denominator. In this fashion, we create a system of modular

equations (in x_2) for the first step of lifting. Moving on, we consider the next lift.

Two consecutive d_1-power liftings. The effective polynomial after the first step was $(L + c(x_2))^{d_1}$. Let us look at the polynomials obtained before division by p^{d_1}. It was $(pL + pc(x_2))^{d_1} + \langle p^{d_1+1} \rangle$. Composing this with another lift of the same kind, the polynomial has to be of the form $(p(pL + pc(x_2)) + p^2\tilde{c}(x_2))^{d_1} + \langle p^{2d_1+1} \rangle$. This implies that we can as well directly lift $L \mapsto p^2L$, divide by p^{2d_1}, and find the value of $c(x_2) + \tilde{c}(x_2)$. So, Eqn.3.5 can be replaced by the lift $p^{-2d_1}\tilde{f}(p^2L, x_2)$ equalling a d_1-power:

$$L^{d_1} + L^{d_1-1} \cdot u_1(x_2)/p + L^{d_1-2} \cdot u_2(x_2)/p^3 + \cdots + u_{d_1}(x_2)/p^{2d_1-1} \equiv (L + u_1(x_2)/pd_1)^{d_1} \bmod p\,, \tag{3.7}$$

Furthermore, we can write down the univariate modular equations like Eqn.3.6.

In this way any i-length contiguous chain of d_1-power liftings, can be directly written as a system of univariate modular equations like Eqn.3.6. It comes from the constraint that the lift $p^{-id_1}\tilde{f}(p^iL, x_2)$ has to equal a d_1-power mod p. Next, this system can be solved by adapting [BLQ13] (see Algorithm 2) to get the representative roots for x_2 variable. Of course, on substituting this in x_2, we will know the final d_1-power L'^{d_1} that the i many lifts yield.

How many consecutive d_1-powers? The length of this chain can be at most k/d_1. So, we go over all $i \leq \lfloor k/d_1 \rfloor$. Iterating over them in decreasing order, we find all the possible ways of getting d_1-powers (before moving to other cases). This ensures that we do not miss any $(\mathbb{Z}/p^k\mathbb{Z})$-root of f in the search-tree.

This can be illustrated through the following examples.

Example 3.10. *Consider the polynomial $f(x_1, x_2) = x_1^2 \bmod p^k$. The d_1-power contiguous chain will be of length $k/2$; and each time $L = x_1$. The corresponding root will be $(p^{k/2} \cdot *_1, *_2)$.*

Example 3.11. *We use a slightly more complicated polynomial this time, with $f(x_1, x_2) = (x_1 + x_2)^2 + p(2(x_1 + x_2)x_2 + px_2^2) + p^6h(x_1, x_2)$. Here the contiguous chain of d_1-power liftings has length 3. The first linear polynomial is $L := x_1 + x_2$*

and the base-change gives $\tilde{f}(L, x_2) = L^2 + p(2Lx_2 + px_2^2) + p^6\tilde{h}(L, x_2)$.

The polynomial after another lifting will be $(L+x_2)^2+p^4\tilde{h}(pL, x_2)$. *Now, the new linear form will be* $L' := L + x_2$. *This we can continue lifting, keeping* L' *unchanged, for* 2 *more steps.*

Notation for x_2 representatives. A problem arises when the representative for x_2 is $*_2$, i.e. x_2 can take any p-adic value. Eg. if we lift $f = L^{d_1} + p^{d_1}x_2^{d_1+1}$ (with free $x_2 = *_2$) then we get $g := L^{d_1} + x_2^{d_1+1}$. The degree of the new polynomial has now *increased*, which we never want to happen in our tree branchings. In order to prevent this, we preprocess the representative root $x_2 = *_2$ by increasing the precision by one coordinate, and thus increasing the number of representative roots by a factor of p (hence, more branchings in the tree). In other words, we consider the representative as $a + p \cdot *_2$, for $a \in \{0, \ldots, p-1\}$.

The following lemma shows that the effective degree never grows in lifting steps, in our algorithm.

Lemma 3.12 (Degree invariant)**.** *The effective degree in each transformation described for* d_1*-forms is always* d_1.

Proof. Let us follow the above notation in the basis $\{L, x_2\}$ and start with effective degree d_1. We now know that every lifting step looks like the map: $L \mapsto pL$ and $x_2 \mapsto a_2 + p^{i_2}x_2$ (for some $i_2 \geq 1$ and integer a_2), followed by division by p^{d_1}. As we calculated in Theorem 3.2, such a lifting step yields effective degree $\leq d_1$. Since we are in the d_1-form case, this implies that the effective degree remains d_1 always. □

Summing up. The structure discovered above gives a natural pseudocode that we describe in Algorithm 4. The contiguous val-mult=d_1 chain will have some d_1-powers, say i_1 many, followed by i_3 many d_1-nonpower forms, from which we have $i_1 + i_3 \leq k/d_1$. The d_1-nonpower forms can not lead to d_1-powers again, due to Lemma 3.9. Also, to get the i_1 many d_1-powers, we need to use Algorithm 2 and get representatives R_1 for x_2 (in general L_2, independent of L_1). Going over each $i_1, i_3 \leq \lfloor k/d_1 \rfloor$, and each of the representatives R_1, we compute the intermediate

representative-roots R (and could continue with our recursion on the local roots with subsequent degree-reduction). This algorithm makes sure that we 'jump' the cases of val-mult=d_1= effective-degree quickly and reach the degree-reduction branchings of the tree.

3.4 Create-Walk( ) subroutine: Completion of the algorithm

In Algorithm 3, since we are iterating over all the possible roots ($O(p^2d)$-many), we do not want the lifting to go on for several steps, since the time complexity is exponential in the number of steps (=tree-depth). The favorable case is when the effective-degree reduces, e.g., when the val-multiplicity of local root is $< d_1$ (from Theorem 3.2). As we will see, once we organize all the possible branches in the tree as portrayed in Figure 3.1, which is a modified branching w.r.t. all possible val-multiplicities, the effective degree will reduce at each level when we go down the tree. This is what the CREATE-WALK() subroutine will achieve inside the red nodes. In this section we sketch the pseudocode based on the ideas developed in Section 3.3.1.

3.4.1 Data-structure returned

In order to lift in such a way, we return an array of tuples of the form $(a_1+p^{u_1}L_1, a_2+p^{u_2}L_2, u_3)$. This gives us information on jumping over the val-multiplicity d_1 roots by first covering the d_1-powers followed by d_1-nonpowers. This is done in a basis (L_1, L_2) of variables possibly different from (x_1, x_2). As in Equation 3.7, we form equations in terms of L_2 and find the roots, such that after lifting according to these (representative) roots, the effective polynomial will be $L_1^{d_1}$. Note that in each lifting according to the fixed part of the representative root, the linear polynomial will change by only a constant. Therefore, $\{1, L_1, L_2\}$ will also span the same space as that of $\{1, x_1, x_2\}$. So, given a root in (L_1, L_2) basis, we can recover the root in

(x_1, x_2) basis uniquely.

With information from this tuple, we can do the following sequence of liftings in 'one-shot': i_1-steps of d_1-powers, followed by i_3-steps of d_1-nonpower-forms.

3.4.2 The algorithm

Summing up, CREATE-WALK( ) in Algorithm 4 is a subroutine used in ROOT-FIND( ) in order to jump over the intermediate local roots of val-multiplicity d_1 without invoking recursive-branching. After this, we find the local roots of val-multiplicity $< d_1$ and continue recursively to degree reducing cases in ROOT-FIND( ) (Steps 6-10 of Algorithm 3).

The *input* is the polynomial with and the prime-power, while the *output* is a tuple of linear polynomials, denoting intermediate representative-roots (over $(\mathbb{Z}/p^k\mathbb{Z})^2$), and length of the d_1-forms chain covered.

Algorithm 4 Finding consecutive intermediate val-mult=d_1 roots in one-shot

1: **procedure** CREATE-WALK($f(x_1, x_2), p^k$)
2: Define $d_1 := \deg(f \bmod p)$, $R := \phi$.
3: **for** $i_1 \in \{\lceil k/d_1 \rceil, \ldots, 0\}$ **do**
4: $R_1 := \phi$
5: Find the linear polynomial L such that $f \equiv L^{d_1} \bmod p$. If L is x_2-multiple, then set $L_2 := x_1$, otherwise set $L_2 := x_2$.
6: Compute the (basis-change) polynomial $\tilde{f}$ such that $\tilde{f}(L, L_2) = f(x_1, x_2)$.
7: Write $\tilde{f}$ as in Equation 3.7 and form (univariate, modular) equations like Equation 3.6 in terms of polynomials in L_2 such that i_1-many contiguous d_1-powers exist (Section 3.3.1).
8: Find the representative-roots, in $\mathbb{Z}/p^{i_1 d_1}\mathbb{Z}$, of the system of equations formed in terms of L_2 (as in the previous step) using Algorithm 2 and store them into R_1.
9: **for** each representative-root $r_2 + p^{i_2} * \in R_1$ **do**

10: Find linear polynomial L_1 obtained in the end, by substituting the representative in L_2, using the method of Section 3.3.1. Note: $i_2 \geq 1$ and L_1 has to be of the form $L + c$ for some integer c.

11: Write $\tilde{f}(L, L_2)$ in basis L_1, L_2 given by $g(L_1, L_2) := \tilde{f}(L, L_2)$.

12: Lift $g(L_1, L_2) := p^{-i_1 d_1} \cdot g(p^{i_1} L_1 \, , \, r_2 + p^{i_2} L_2)$

13: **for** $i_3 \in \{ \lceil k/d_1 \rceil - i_1, \ldots, 0 \}$ **do**

14: **if** $\exists \mathbf{r'} \in (\mathbb{Z}/p^{i_3}\mathbb{Z})^2$ s.t. g is a d_1-nonpower-form consecutively i_3-times **then**

15: In each precision $\mathbf{r'}$ is unique; so it can be searched easily in the space $\mathbb{F}_p^2$.

16: $R_0 := (p^{i_1} r'_1 + p^{i_1+i_3} L_1 \, , r_2 + p^{i_2} r'_2 + p^{i_2+i_3} L_2 \, , i_1 + i_3 \,)$

17: **if** $R_0 \notin R$ **then**

18: $R := R \cup \{R_0\}$

19: **else**

20: **break**

21: **return** R

3.4.3 Proof of correctness of Algorithm 3

We prove the correctness of our root finding algorithm (Algorithms 3 and 4) in the following theorem.

Theorem 3.13 (Correctness of Algorithm 3)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$ of degree d, a prime p and an integer k. Algorithm 3 using Algorithm 4 as a subroutine, correctly returns all the roots $(a_1, a_2) \in (\mathbb{Z}/p^k\mathbb{Z})^2$ of $f(x_1, x_2) \bmod p^k$, in deterministic $\mathsf{poly}((k + d + p)^d)$ time.*

Proof. In this proof, we will analyze the structure of the roots-tree given by Figure 3.1. As described in Sections 3.1 and 3.4, all the green nodes in this tree are in degree-reducing cases since we are calling the recursion to the next step using Root-Find( ) only on val-multiplicity $< d_1$ roots (Steps 9 & 21 of Algorithm 3).

Thus, whenever we go down this tree by iterating over all the roots of val-multiplicity $< d_1$, effective degree d_1 decreases by at least 1. Here, we fix one variable to any value in $\{0, \ldots, p-1\}$, and find the p-many possibilities of the other variable. The red node of Figure 3.1 also gives val-multiplicity $< d_1$ roots after two levels in the tree. Thus, the depth of this tree is at most $2d$, after which we reach the *leaves*. At a leaf, either the precision of p^k has been reached *fixing* a root, or the effective polynomial has become *linear*, or *no* root exists.

For the width/fanin of the tree, the number of possible lengths of d_1-powers is $\lceil k/d_1 \rceil = O(k)$, while that of d_1-nonpower-forms is also $O(k)$. Therefore, the number of possibilities of lengths of val-mult$=d_1$ branches is $O(k^2)$. In Algorithm 4, all the val-mult$=d_1$ branches of length $i_1 + i_3$ for i_1-length contiguous d_1-powers and i_3-length contiguous d_1-nonpower-forms are returned. As we have seen in Section 3.3.1, we can solve for the intermediate values of L_2, from which we find the appropriate L_1 too, such that d_1-powers are possible. After this, we come straight down to the d_1-nonpower-forms. Furthermore, from Theorem 2.20 and Equation 3.6, the number of representatives for L_2 is $O(d)$, which may become $O(pd)$ after preprocessing to handle $*$.

This bounds the number of roots returned by CREATE-WALK(f, p^k) by $O(k^2dp)$. In a deterministic version of Algorithm 2, we exhaustively search for every L_2 root at each step, so its complexity is $\mathsf{poly}(kdp)$-time. Therefore, all the operations inside CREATE-WALK() can be performed in $\mathsf{poly}(kdp)$-time.

Since the depth of the tree is $2d$ (in Fig.3.1), the total size of the tree gets bounded by $\mathsf{poly}((k+d+p)^d)$. Since all other operators are usual field operations and search in $\mathbb{F}_p$, the net time complexity of our algorithms is just a polynomial overhead of going over all these nodes. This proves the complexity to be deterministic $\mathsf{poly}((k+d+p)^d)$-time. □

Proof of Theorem 1.1. Algorithm 3, using Algorithm 4 as a subroutine, correctly returns all the roots of $f(x_1, x_2) \bmod p^k$, in a representatives form, in deterministic $\mathsf{poly}((k+d+p)^d)$ time (due to Theorem 3.13)

By the time the algorithm terminates, we will have $O((k^2dp)^d)$ leaves of the tree, where we have either found roots or encountered a dead-end. If roots exist, then each such leaf signifies that either

1. we reached p^k and a fixed root (a_1, a_2) is returned,

2. or, we reached p^k and (some invertible linear transformation of) the representative $(p^{i_1} \cdot *_1, p^{i_2} \cdot *_2)$ is returned. The total number of roots in this case is $p^{2k-i_1-i_2}$,

3. or, when the effective polynomial becomes linear and $(*, c(*))$ is returned (or some invertible linear transformation of it); where $*$ represents all values in $\mathbb{Z}/p^{k-i}\mathbb{Z}$. Here, we fix only one variable to find the value of the other at each step for $k - i$ Hensel lifts, implying that the number of roots is p^{k-i}.

Using this technique to sum over all the leaves which do not lead to dead-ends, we can count the total number of roots in deterministic $\mathsf{poly}((k + d + p)^d)$ time. □

3.5 Generalization to n-variates

In this subsection, we generalize our approach to root-finding and counting to n-variate polynomial $f(\mathbf{x})$ using similar techniques as used in bivariates, and reducing the problem to finding roots of $(n-1)$-variate polynomial systems. In order to do so, we first show a modification to Algorithm 3 in order to solve a system of polynomial equations $\bmod p^k$. This subsection gives an overview of how to extend the single bivariate root-finding algorithm to that for solving a *system* of bivariates. Then, using that how to solve 3-variate systems. The proof can be straightforwardly generalized to n-variates, for any $n \geq 3$.

Solving bivariates simultaneously. Suppose we have m polynomials $f_1(x_1, x_2), \ldots, f_m(x_1, x_2)$, each of degree $\leq d$. At each step of lifting, we iterate over all the roots of each polynomial separately, but in parallel, such that the local roots of each iteration are common to every polynomial. This can be intuitively thought

of as creating trees like Figure 3.1 corresponding to each polynomial, whose nodes are 'isomorphic', i.e. we branch corresponding to a root if and only if the root is present in the trees of all the polynomials. Also, whatever linear transformation we apply on $\mathbf{x}$ acts on all the f_i's simultaneously. Thus, we refer to this process as *parallel* search-tree.

We continue growing these trees (and considering only those branches which correspond to a common root); with effective degree decreasing as we go down, until the stopping conditions are obtained. When some polynomial has a representative root $(*_1, *_2)$, then we can proceed to finding roots of the remaining set of polynomial. However, when linear-representative roots are obtained for some polynomial, the analysis can be divided into two cases by their *rank*. When the linear forms (i.e. given by coefficients of x_1 and x_2) are of rank 1 or 2. For rank 2, we can check for a unique root by solving simple linear equations; so, they either have one common root, or none.

The difficulty arises when these linear forms are of rank 1. Again, the isomorphism of the trees corresponding to each polynomial will be used, but after reducing this problem to solving simultaneous equations over lesser number of polynomials.

Solving bivariates– rank= 1 linear forms. Suppose we have the polynomials in the form $ax_1+bx_2+c_i+ph_i(x_1, x_2)$, where $a, b, c_i \in \{0, \ldots, p-1\}$, for all $i \in [m]$. If all the c_i's are not the same, we obviously do not have a solution; so we terminate this branch. Assume $c_i = c$ and write the polynomials in the basis of $L := (ax_1 + bx_2 + c)$ and x_2, w.l.o.g assuming $a \neq 0$. (Otherwise, we can use the basis L, x_1). We have the system as

$$L \equiv pg_1(L, x_2) \bmod p^k\,;\ L \equiv pg_2(L, x_2) \bmod p^k\,;\quad \ldots\ ;\ L \equiv pg_m(L, x_2) \bmod p^k\,, \tag{3.8}$$

where g_i's can be obtained from h_i's by using the change of basis. So, we get the local-root here, namely L must be 0 mod p in the current step (based on any value of x_2).

Hence we lift L to pL. This grows the tree further. Performing this transformation and subtracting the first equation from each of the other equations, we have :

$$L \equiv g_1(pL, x_2) \bmod p^{k-1} \; ; \; 0 \equiv \tilde{g}_2(pL, x_2) \bmod p^{k-1} \, ; \quad \ldots \; ; \; 0 \equiv \tilde{g}_m(pL, x_2) \bmod p^{k-1} \, , \tag{3.9}$$

where $\tilde{g}_i = g_i(L, x_2) - g_1(L, x_2)$, for $i \in \{2, \ldots, m\}$.

Now, on the $(m-1)$ $\tilde{g}_i$, we apply another instance of Algorithm 3, in a parallel way. For a fixing of x_2 from the latter $m-1$ equations in Equation 3.9, we uniquely get the value of L from the first equation (where the effective polynomial is linear in L). Using these values, we lift both L and x_2, creating a Figure 3.1-like tree where the branches are such that they satisfy all the m equations. Finally, at the leaf we get the representatives for x_2 too, and halt the algorithm.

Thus, we create the m isomorphic trees for $O(d)$ many lifting steps as before (Figure 3.1), then restrict the linear condition on the first polynomial and simultaneously solve the next $m-1$ polynomials modulo a smaller power of p and continue with our construction of $m-1$ isomorphic trees. Using this subroutine, we can find all the roots of the system of bivariate equations in time complexity same as that of Theorem 1.1, along with a multiplicative overhead of m for storing this array of polynomials in each node.

Solving 3-variate— Lifting Step. Given a root $\mathbf{a} \in \mathbb{F}_p^3$ and polynomial $f_j(\mathbf{x})$ in the j-th step, we find the polynomial after lifting as $f_{j+1}(\mathbf{x}) = p^{-v} f_j(\mathbf{a} + p\mathbf{x})$, where $v = v_p(f_j(\mathbf{a} + p\mathbf{x}))$ is the val-multiplicity of the root $\mathbf{a}$. Theorem 3.2 can be similarly proved to show that effective degree reduces in all cases other than when $d_1 = 1$ or $v = d_1$. We again form a tree similar to Figure 3.1 with the invariant that effective degree must reduce along depth.

Solving 3-variate— Val-multiplicity d_1 case. As in Algorithm 4, we again 'jump' over the val-multiplicity d_1 cases directly so that the recursion can continue to degree reduction cases. Again, Lemma 3.7 can be proved for 3-variates to show that the polynomial must have the d_1-form $\langle x_1 - a_1, x_2 - a_2, x_3 - a_3 \rangle^{d_1}$, where $\mathbf{a}$ is

a val-multiplicity d_1 root. However, when multiple val-multiplicity d_1 roots exist, w.l.o.g. given by $\mathbf{0}$ and $\mathbf{a}$, where $a_1 \neq 0$, we can modify the proof of Lemma 3.8 to show that the effective polynomial is zero modulo $\langle a_1x_2 - a_2x_1, a_1x_3 - a_3x_1\rangle^{d_1}$. The rank of the nonzero roots, given by $\langle \mathbf{x} - \mathbf{a}\rangle$, can be 1 or 2 (in general, $1, 2, \ldots, n-1$ for n-variates).

In the case of rank=2 val-mult=d_1 roots, there will be only one linear polynomial, say $f \in \langle a_1x_2 - a_2x_1\rangle^{d_1} + \langle p\rangle$. So, the equations will be like Eqn.3.5, except that the polynomials u_j's will be in two variables, say x_2 and x_3. We will form a bivariate system, and solve it using the simultaneous bivariate root-finding as discussed above; to find all the representative-roots for x_2, x_3. This new version of Algorithm 4 will take $\mathsf{poly}((k+d+p)^d)$-time; and make the tree this much wider.

In the case of rank=1 val-mult=d_1 roots, we will have a multinomial expansion having two linear forms $\{a_1x_2 - a_2x_1, a_1x_3 - a_3x_1\}$ instead of a single binomial expansion as done in Equation 3.7. So, now, we have two linear forms L, L' instead of a single L_1. We need to find the values of the third variable, say x_3, such that the resulting effective polynomial after lifting is again in $\langle L, L'\rangle^{d_1}$. This gives constraints similar to Equation 3.6. From these, we can use the univariate Algorithm 2 to solve them.

Finally, the techniques of Section 3.3.1 can be smoothly generalized to search for the contiguous d_1-forms in $\mathsf{poly}((k+d+p)^d)$-time. In particular, we will search them in the decreasing order of the rank of underlying val-mult=d_1 roots: rank=2, rank=1 and then rank=0 in the last.

Conclusion. Using these techniques, we can find the roots of $f(\mathbf{x}) \bmod p^k$, for 3-variates where the only difference from bivariates is the handling of contiguous val-multiplicity d_1 roots (due to the more possibilities of d_1-forms). The same extension can be performed for n-variate m polynomials, for any constant n. Thus, Algorithm 3 fits well in the general framework, and finds all the roots.

Time complexity. For a 3-variate polynomial f, at each step of the tree (Fig.3.1), there are $O(dp^3)$ branches corresponding to val-mult$< d_1$ roots. For val-

mult=d_1 roots, there are three ordered possibilities in the chain: rank=2 val-mult=d_1 root, rank=1 val-mult=d_1 roots, and rank=0 (unique) val-mult=d_1 roots. Similar to Lemma 3.9, we can show that rank can not increase after lifting by a val-mult=d_1 root.

For rank=2, we solve a system of bivariate equations. The number of possible branches of bivariates is $O((k^2d+p^2)^{2d})$. Furthermore, the number of possibilities of these special contiguous chains is $O(k^3)$. Therefore, the total number of leaves of the tree for 3-variates will be $O(\,((k^2d+p^2)^{2d}\cdot k^3+p^3)^d\,)$, which is bounded by $O((k+d+p)^{(4d)^2})$. Assume, for induction hypothesis, that the time complexity, for any n, is bounded by $O((k+d+p)^{(2d(n-1))^{n-1}})$.

For n-variates ($n\geq 3$), the val-multiplicity d_1 roots will have ranks from $n-1$ to 0. For rank=$(n-1)$, we get the maximum upper bound. We will be solving a system of $(n-1)$-variate equations, which will lead to $O((k+d+p)^{(2d(n-2))^{n-2}})$ possibilities and representative-roots. Furthermore, there are k^{n-1} possibilities for the chain. Thus, one tree size is $s_1 := O(\,(k^{n-1}(k+d+p)^{(2d(n-2))^{n-2}}+p^n)^{2d}\,)$. But we may need to repeat this $n-1$ times on each leaf, when we have a system of n-variates to solve. Thus, the time becomes $s_1^{n-1}=O((k+d+p)^{(2d(n-1))^{n-1}})$.

Using this technique for finding all the roots modulo p^k for n-variates, we can generalize the algorithms for finding $\mathbb{Z}_p$-roots and computing the Igusa local zeta function for a system of m polynomials in n-variables as well. For finding roots in $\mathbb{Z}_p$, we consider the resultant w.r.t. one variable at a time, find a bound similar to Lemma 4.4, and proceed with the analysis of roots which are in $\mathbb{Z}_p$ or are not roots of the discriminant. At each step, the bounds according to one variable at a time will be obtained, which get multiplied to give a bound k_0 for $f(\mathbf{x})$ such that roots of $f(\mathbf{x})$ mod p^{k_0} gives us roots which correspond to $\mathbb{Z}_p$ roots as well. This will be a generalization of Theorem 4.10 to n-variates.

Chapter 4

Computing the Igusa's Local Zeta Function

We use the results of Chapter 3 to first show give a bound k_0 depending on the polynomial $f(x_1, x_2)$ such that roots of $f(x_1, x_2)$ over $\mathbb{Z}/p^{k_0}\mathbb{Z}$ correspond exactly to the roots of $f(x_1, x_2)$ over $\mathbb{Z}_p$. Namely, we prove the following two Corollaries as consequences of Theorem 1.1.

Corollary 4.1 (p-adic)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can find all the p-adic-roots of f (in $\mathbb{Z}_p$) in deterministic $\mathsf{poly}((\log M + d + p)^d)$ time (i.e. output their k digits in a compact data structure).*

Corollary 4.2 (Local-zeta fn.)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can compute the Poincaré series $P(t) =: A(t)/B(t)$ associated with f and a prime p, in deterministic $\mathsf{poly}((\log M + d + p)^d)$ time.*

4.1 Describing the roots over $\mathbb{Z}_p$

In this subsection, we give a bound for k_0 in terms of the degree d and the maximum absolute value M of the coefficients, such that finding a root modulo p^{k_0} would imply

finding all representative $\mathbb{Z}_p$-roots of f.

4.1.1 Preprocessing: Reduce to radical case

We are concerned with the roots of $f(x_1, x_2)$ in $\mathbb{Z}_p$, which is an integral domain (with fraction field $\mathbb{Q}_p$). The polynomial will have a unique factorization in $\mathbb{Z}_p$, which will be of the form

$$f(x_1, x_2) \;=\; \prod_{i=0}^{r} g_i(x_1, x_2)^{e_i} \;, \tag{4.1}$$

where $g_i(x_1, x_2)$'s are coprime over $\mathbb{Z}_p$. Even if f has some square-full factors (some e_i's are ≥ 2), we can eliminate them efficiently, by computing its gcd with the first-order derivatives. This will result in the new polynomial being of the form $\prod_{i=0}^{r} g_i(x_1, x_2)$, which we will call the *radical polynomial* $\mathrm{rad}(f)$. The polynomial f and its radical $\mathrm{rad}(f)$ have the same set of roots in $\mathbb{Z}_p$. In the following lemma, we bound the absolute-value of the coefficients of the radical, by M^d.

Lemma 4.3 (Bound for p-adic radical)**.** *If a polynomial f of degree d has the absolute-value of its coefficients bounded by M, then its radical has its coefficients bounded by $M^{O(d)}$.*

Proof. We prove this using Euclid's algorithm for finding gcd, when we consider the gcd of f and any one of its first-order derivative f'.

At each step of Euclid's gcd algorithm, we have two polynomials q_i and q_{i+1}, where $\deg(q_i) \geq \deg(q_{i+1})$. We compute the remainder of q_i when divided by q_{i+1}, assume it to be q_{i+2}, and then proceed to do the same with q_{i+1} and q_{i+2}.

Now, it can be inductively shown that the coefficients of q_i is bounded by M^{F_i}, where F_i is the i-th Fibonacci number. This is true as while dividing q_{i-2} by q_{i-1}, the quotient will have its coefficients bounded by that of q_{i-2}. This quotient multiplied by q_{i-1} will give the bound for the remainder, which thus is bounded by the product of bounds of coefficients of q_{i-1} and q_{i-2}. Now, this procedure continues for $\log d$ steps, implying that the coefficients of the gcd of f and its derivative is bounded by $M^{F_{\log d}} = M^{O(d)}$. Dividing f by this gcd will give the bounds on the coefficient of

$\text{rad}(f)$, which is also $M^{O(d)}$. □

Therefore, w.l.o.g. we consider $f(x_1, x_2)$ to be square-free having absolute value of coefficients $\leq M^{O(d)}$, and continue with our algorithm of finding roots in $\mathbb{Z}_p$.

4.1.2 Representative roots in $\mathbb{Z}/p^k\mathbb{Z}$ vs roots in $\mathbb{Z}_p$.

The return value of the algorithm, in the base-case, is either the representative root $(*_1, *_2)$ when the exponent of p required gets achieved (Step 2 of Algorithm 3), or linear-representative root $(*, c(*))$ (Steps 4-5 of Algorithm 3).

For large enough k, i.e. $k > k_0$, we want to show that if a representative root $(*_1, *_2)$ is returned, then the fixed part of the root is already a $\mathbb{Z}_p$-root. In the other case, for linear-representative roots, we can simply use Hensel lifting to lift to $\mathbb{Z}_p$-roots (or, to as much precision as we wish).

4.1.3 Bound to distinguish $\mathbb{Z}_p$ roots

Consider $R(x_1) = \mathsf{Res}_{x_2}(f(x_1, x_2), f'(x_1, x_2))$, the discriminant w.r.t. x_2. The roots of $R(x_1)$ given by $\hat{x}_1$ satisfy the condition that the univariate $f(\hat{x}_1, x_2)$ is square-full. Furthermore, given $\hat{x}_1$, we can easily find the values of x_2 (d-many), as it becomes the univariate root-finding problem over $\mathbb{Z}_p$ which has a famous solution (see Algorithm 2).

The main idea is to find a bound for the exponent of p such that each root returned using root-finding is either a linear-representative root, or a unique lift of this root is a $\mathbb{Z}_p$ root. A similar bound was achieved for univariate polynomials by [DS20]. However, the complications of lifting multivariate roots did not arise there, as every p-adic root corresponded to a p-adic factor.

Lemma 4.4 (Discriminant of radical)**.** *Let $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$ be a polynomial of degree d whose coefficients have absolute value bounded above by M. Let its radical polynomial be $g := \text{rad}(f)$. The roots of $R(x_1) = \mathsf{Res}_{x_2}(g, g')$ in $\mathbb{Z}_p$ are in one-one correspondence to the representative roots of $R(x_1) \bmod p^k$, for any $k \geq k_1 :=$*

$\Theta(d^6 \log M)$.

Proof. We have the polynomial $f(x_1, x_2)$ of degree d. Its radical polynomial $g := \text{rad}(f)$, has degree $\leq d$ and coefficients bounded by $M^{O(d)}$ (Lemma 4.3).

The resultant polynomial $R(x_1) = \mathsf{Res}_{x_2}(g, g')$ is the determinant of a $(2d+1) \times (2d+1)$ matrix consisting of elements formed from the coefficients of g. Thus, the degree of $R(x_1)$ is $< 2d^2+d$, and the absolute-value of the coefficients is $< (dM^d)^{2d+1}$.

Now, we need to find a bound on k_1 such that the roots of $R(x_1)$ are in one-one correspondence to those of $g(x_1, x_2)$ mod p^{k_1}. [DS20, Theorem 20] showed that the representative roots of a univariate polynomial modulo p^k, for $k > d'(\Delta + 1)$, are in one-one correspondence to the roots of that polynomial in $\mathbb{Z}_p$, where d' is the degree and Δ is the p-valuation of its discriminant. In our case of finding $\mathbb{Z}_p$-roots of $R(x_1)$, the degree is $2d^2 + d$, while the discriminant is at most $((dM^d)^{2d+1})^{2(2d^2+d)+1}$. Thus, the valuation of discriminant of R is bounded by $O(d^4 \log_p M)$. Substituting the values of d' and Δ, we have $k_1 := \Theta(d^6 \log_p M)$. □

4.1.4 $\mathbb{Z}_p$-roots

Consider $g = \text{rad}(f)$ and $k_1 = \Theta(d^6 \log M)$. Define $g_2(x_2) := \text{Res}_{x_1}(g(x_1, x_2), R(x_1))$. Intuitively, roots x_2 of g_2 come from the roots x_1 of R. So, again applying [DS20, Theorem 20] on this univariate polynomial g_2, it suffices to compute its roots mod p^{k_2}, to compute its distinct p-adic roots; where k_2 is asymptotically $\log_p(p^{k_1 \cdot 2d^2 \cdot d})$ $= O(d^9 \log M)$.

Using the value of k_2 as above, we find roots of $g(x_1, x_2)$ from $\textsc{Root-Find}(g, p^{k_2})$. Let $(\tilde{a}_1, \tilde{a}_2)$ be the fixed-part of a root thus obtained. If $R(\tilde{a}_1) \equiv 0 \bmod p^{k_2}$, then the above argument, that defined k_2, ensures that $(\tilde{a}_1, \tilde{a}_2)$ does lift to a $\mathbb{Z}_p$-root of R, g_2, g and f (in this case uniquely).

4.1.5 Non-root of discriminant.

Thus, the case left is: $R(\tilde{a}_1) \neq 0 \bmod p^{k_2}$. Consider the univariate $g(\tilde{a}_1, x_2)$. We know that its $\mathbb{Z}_p$-roots are different mod p^{k_2} and at most d many; one of which is $\tilde{a}_2$.

Consider $g_1(x_2) := p^{-v} \cdot g(\tilde{a}_1\,,\, \tilde{a}_2 + x_2)$, where $v \geq 0$ is the p-valuation of $g(\tilde{a}_1\,,\, \tilde{a}_2 + x_2)$ as a polynomial over $\mathbb{Z}_p$. Note that x_2 divides g_1, but x_2^2 does not divide $g_1 \pmod{p}$. Thus, 0 is a simple-root of g_1 and we can potentially Hensel lift it to p-adics.

To implement this formally, we need to increase the precision so that the extra p-factors can be removed from g. Note that if we assume $p \nmid g(\tilde{a}_1, x_2)$ then $v \leq k_2 + (k_2 - 1)(d-1) < d \cdot k_2$. Fix $k_0 := d \cdot k_2 = \Theta(d^{10} \log M)$. Now consider $\tilde{g}(\mathbf{x}) := p^{-v} \cdot g(\tilde{a}_1 + p^{k_2} x_1\,,\, \tilde{a}_2 + p^{k_2} x_2) \bmod p^{k_0}$. By the argument above, $\tilde{g} \bmod p$ is linear in x_2 (it is easier to see by substituting $x_1 = 0$). Thus, an extension of this root has to end up in some leaf of ROOT-FIND(g, p^{k_0}) algorithm as say $(\tilde{a}_1', \tilde{a}_2')$; which will Hensel lift to p-adic integral root(s) due to the linear x_2 term in the lift.

Since the set of p-adic roots for f and g is the same, we could as well run ROOT-FIND(f, p^{k_0}). This proves the following lemma.

Lemma 4.5 (p^{k_0} is p-adic)**.** *Given a polynomial $f(x_1, x_2) \in \mathbb{Z}[x_1, x_2]$ of degree d and having absolute-value of coefficients bounded by M. Each root represented in the leaves of the tree of ROOT-FIND(f, p^{k_0}), for $k_0 := \Theta(d^{10} \log M)$, lifts to a $\mathbb{Z}_p$-root of $f(x_1, x_2)$.*

We further need the condition that the structure of this tree does not change with k for $k \geq k_0$. In order to show that, we prove the following lemma. Denote $R_1(x_1) := \mathsf{Res}_{x_2}(g, \partial_{x_2}(g))$ and $R_2(x_2) := \mathsf{Res}_{x_1}(g, \partial_{x_1}(g))$

Lemma 4.6 (Fix p-adic tree)**.** *If a leaf of the tree given by Lemma 4.5 returns a representative root with the fixed part (a_1, a_2), that is not linear-representative, then $R_1(a_1) = R_2(a_2) = 0 \bmod p^k$. Moreover, (a_1, a_2) lifts to a unique root of f over $\mathbb{Z}_p$; and their number does not change as k grows beyond k_0.*

Also, the tree (Fig.3.1) in our algorithm does not change, and remains isomorphic, for $k \geq k_0$; except the leaf with the root $\mathbf{0}$.

Proof. As argued above, the representative roots which are not linear-representatives, must satisfy the condition $R_1(a_1) \equiv 0 \bmod p^{k_0}$. Using a similar technique we can show that $R_2(a_2) \equiv 0 \bmod p^{k_0}$ as well.

Assume that for some large enough k, $k \geq k_0$, a new leaf in the tree of Figure 3.1 appears, with the root (r_1, r_2) such that $R_1(r_1) \equiv R_2(r_2) \equiv 0 \bmod p^k$. However, this leads to a contradiction as the branch corresponding to (r_1, r_2) should have already been present in the tree at precision k_0 in the representative root.

As argued before, this root r_j of $R_j \bmod p^{k_0}$ always lead to $\mathbb{Z}_p$ roots of R_j for $j = 1, 2$ (due to [DS20, Theorem 20]), and that of g_2, g, f. Thus, the number of representative roots can not decrease. Therefore, the number of representative roots which are not linear is fixed once we reach k_0, and hence (a_1, a_2) has a unique lift to $\mathbb{Z}_p$.

Together with Hensel lifting, it is then clear that, the linear-representative roots can neither increase in number, nor reduce, as $k \geq k_0$ grows.

The only change that could happen is, for $k \geq k_0$, if the leaf with fixed root $\mathbf{0}$ is used to lift to $f(p^v x_1, p^v x_2) \bmod p^k$, with $k > v \geq k_0$. This may create a new subtree under the old leaf $\mathbf{0}$; as this type of branches are the only ones that were not explored in Algorithm 3 $\bmod p^{k_0}$. □

The following examples should help illustrate the p-adic machinery more clearly.

Example 4.7. *Consider the polynomial* $f = (x_1 - 1)(x_2 - 2) \bmod p^k$. *The first step of our algorithm has to be* $x_1 = 1$ *or* $x_2 = 2$. *Considering the root* $\mathbf{a} := (1, 3)$, *the polynomial after lifting becomes* $x_1(1 + px_2)$, *which is an (effective) linear form; thus, a linear-representative root will be returned, which has* x_2 *as the free variable while* x_1 *will stay fixed to* 1. *This gives the leaf* $\mathbf{r} := (1 + p\mu(*), 3 + p*)$, *and a computable* $\mathbb{Z}_p$*-function* $\mu(\cdot)$, *which allows the p-adic lift of* $\mathbf{a}$. *In this case,* $\mu = 0$.

Example 4.8. *Now, consider* $f(x_1, x_2) = (x_1 - px_2)(x_1 - 2px_2) \bmod p^k$. *Lifting the root* $\mathbf{a} := (0, 1)$ *gives us* $(x_1 - 1 - px_2)(x_1 - 2 - 2px_2)$, *which is not yet effective linear. Choosing the next lifting-step around the root* $(1, 0)$, *the polynomial after lifting becomes* $(x_1 - px_2)(-1 + px_1 - 2p^2x_2)$, *which is an (effective) linear polynomial; thus, a linear-representative root will be returned, corresponding to* $(x_1 - px_2)$, *which has* x_2 *as the free variable while* x_1 *depends on it. This gives the leaf* $\mathbf{r} := (p + p^3\mu(*),\ 1 + p^2*)$,

and a computable $\mathbb{Z}_p$-function $\mu(\cdot)$, which allows the p-adic lift of $\mathbf{a}$. *In this case,* $\mu(w) := w$.

However there are several roots in $\mathbb{Z}_p$, which can not be noticed modulo p^{k_0}, because they are indistinguishable from $\mathbf{0}$. This is seen in the following example.

Example 4.9. *Consider the polynomial $x_1^3 + x_2^3 \bmod p^k$, for $p > 3$ and $3|k$. Some of its linear-representative roots are $(p^j + p^{j+1}*, -p^j + p^{j+1}\mu(*))$, for any $j < k/3$ and $\mu(w) := -w$. Also, $(p^{k/3}*_1, p^{k/3}*_2)$ is a non linear-representative root. It can lift to the p-adic root $\mathbf{0}$, but it can also lift to $(p^k, -p^k)$; which our algorithm could not explore due to the precision being only p^k.*

The following theorem completes the connection, of Algorithm 3, with *all* p-adic roots of f. Basically, it scales up the roots by p^v-multiple, whenever possible, and creates a new data-structure for representatives in the leaves of the fixed tree modulo $\bmod p^{k_0}$, in Fig.3.1. It can also be seen as a way of further *blowing-up* the leaf of the fixed tree that gives the $\mathbf{0}$ root.

Theorem 4.10 (High val p-adic roots)**.** *We can efficiently 'expand' our leaves as follows:*

(1) Define a set of representative-roots $\mathcal{H}_v$, for $v \geq k_0$, s.t. for each root $\mathbf{a} \in \mathcal{H}_v$, $p^v\mathbf{a}$ lifts to a p-adic root of f.

(2) We can compute the fixed tree for $\mathcal{H}_{k_0}$ efficiently by Algorithm 3. The other sets $\mathcal{H}_v$, for $v > k_0$, lift from the same representatives as in the leaves of $\mathcal{H}_{k_0}$; so we do not recompute them.

Let (r_1', r_2') be a p-adic root of f. Then, $\exists v \geq 0$, $\exists$ root $\mathbf{a} \in \mathcal{H}_v$ lifting to $\mathbf{a}'$, for which $(r_1', r_2') = p^v\mathbf{a}'$. In this sense, our fixed tree covers all p-adic roots of f.

Proof. Let u be the p-valuation of $\mathbf{r}'$, i.e. $p^u||(r_1', r_2')$. If $u = \infty$, i.e. $(r_1', r_2') = \mathbf{0}$, then clearly some leaf in the set $\mathcal{H}_{k_0}$ will satisfy the required statement.

If $u < k_0$, then $\mathbf{r}' \neq 0 \bmod p^{k_0}$; so it will be covered in some nonzero leaf of the tree of Lemma 4.6.

Assume $\infty > u \geq k_0$. Now consider the system $f(p^u \cdot \mathbf{x}) = 0$, say over $\mathbb{Z}_p$. Write $f =: \sum_{m \leq i \leq d} f_i$ into homogeneous-parts, with m being the least-degree part ($f_m \neq 0$). Thus, by homogeneity, the system becomes

$$0 \;=\; f(p^u \cdot \mathbf{x})/p^{um} \;=\; \sum_{m \leq i \leq d} p^{u(i-m)} \cdot f_i(x_1, x_2)\,. \tag{4.2}$$

If $f = f_m$ (i.e. f is homogeneous), then $f(p^u \cdot \mathbf{x}) = 0$ iff $f(\mathbf{x}) = 0$. Thus, $\mathcal{H}_v$, for all $v \geq 0$, is given by the fixed tree in Lemma 4.6 and we are done.

Assume $f \neq f_m$ (i.e. f is *in*homogeneous). Then, the above system implies: $f_m(x_1, x_2) \equiv 0 \bmod p^u$. Since f_m has bounded coefficients and $u \geq k_0$, we compute the fixed tree (Lemma 4.6 for $f_m \bmod p^{k_0}$) efficiently; and all its leaves (except $\mathbf{0}$) lift to p-adic roots. Each leaf, in our Algorithm 3 can be viewed as defining a nontrivial p-adic map $\mu : \mathbb{Z}_p \to \mathbb{Z}_p$ s.t. w.l.o.g., $f_m(w, \mu(w)) = 0$, where w is a variable. Check whether $f(p^u w\,, p^u \mu(w)) = 0$. Then, by repeating this argument (on f_{m+1} etc.) we can deduce: $f_i(w, \mu(w)) = 0$ for all $m \leq i \leq d$. Since, μ is a p-adic function common to these polynomials, that are all upper bounded by the parameters of f, we can learn μ by working with each f_i just $\bmod p^{k_0}$.

Algorithmically, we find this common μ by first invoking Algorithm 3 on $f_m \bmod p^{k_0}$ and then verifying it for $f(p^{k_0} w\,, p^{k_0} \mu(w)) \equiv 0 \bmod p^{k_0(d+1)}$. [Or, we could construct the tree common to the system $\{f_m, \ldots, f_d\} \bmod p^{k_0}$.]

But this shows an interesting property p-adically that $f(p^{k_0} w\,, p^{k_0} \mu(w)) = 0$ iff $f_i(w, \mu(w)) = 0$, for all $m \leq i \leq d$ iff $f(p^u w\,, p^u \mu(w)) = 0$, for all $u \geq 0$. Essentially, the high-valuation roots arise only from homogeneous polynomial system! □

Lemmas 4.5-4.6 and Theorem 4.10 describe the p-adic nature of the tree and the representative roots, after the threshold bound of k_0. This finishes the proof of Corollary 4.1.

4.2 Computing the Igusa's local zeta function

We will show how to compute the Poincaré series, by expressing the number of roots of $f(x_1, x_2)$ mod p^k, for every k, in a special form. In this subsection, we sketch this algorithm for bivariates using Section 4.1. With more work it can be generalized to n-variates based on the machinery of Section 3.5; thus proving the rationality of the Poincaré series in general.

When we consider f modulo p^k, for large enough k's, the fixed-part of the representative-roots will correspond to p-adic roots, while the remaining-part has 'free' coordinates, eg. $(*_1, *_2)$, which get fixed as we increase k. For $k = k_0$, denote $\mathcal{R}$ as the subset of representative-roots which are not linear-representative roots, while the set $\mathcal{L}$ as the set of linear-representative roots.

Recall the bound of $k_0 = \Theta(d^{10} \log M)$ (Lemma 4.5) to distinguish between $\mathbb{Z}/p^k\mathbb{Z}$ and $\mathbb{Z}_p$-roots: For small values of k, i.e. $k < k_0$, we can count the number of roots in deterministic time $\mathsf{poly}((d + p + \log M)^d)$. For large k's, however, we want to prove a special form to sum up the infinite Poincaré series.

4.2.1 Non linear-representative roots $\mathcal{R}$

Consider a root in $\mathcal{R}$; its fixed-part, say $\mathbf{r}$, will lift to $\mathbb{Z}_p$-roots of f. Its representative part appears due to the contribution of extra p-powers by the other derivatives that appear in the Taylor-series around $\mathbf{r}$. Let e be the multiplicity of $\mathbf{r}$: which can be found as the (largest) e such that $f \in \langle \mathbf{x} - \mathbf{r} \rangle^e$, but $f \notin \langle \mathbf{x} - \mathbf{r} \rangle^{e+1}$.

Now, f can be written as

$$f = \sum_{i=0}^{e} c_i(x_1, x_2) \cdot (x_1 - r_1)^i (x_2 - r_2)^{e-i} . \tag{4.3}$$

Define $v := v_p(\gcd(c_i(\mathbf{r}) \mid i))$. Since $\mathbf{r}$ is not a common root of c_i's, this value (e, v) does not change as we increase k and make $\mathbf{r}$ more precise. Consider the $\mathbb{Z}_p$-root $\mathbf{r}'$ that $\mathbf{r}$ lifts to. Let us now consider the ways in which the digits of $\mathbf{r}'$ may be

perturbed, and yet it be a root (mod p^k), as we increase k arbitrarily. Let ℓ_1 denote the length up to which we want to keep r_1 equal to r'_1 and the rest to $*$ (similarly define ℓ_2). Consider

$$f(r'_1+p^{\ell_1}x_1\,,r'_2+p^{\ell_2}x_2) \;=\; \sum_{i=0}^{e} c_i(\mathbf{r}'+p^{\mathbf{l}}\mathbf{x})\cdot(r'_1-r_1+p^{\ell_1}x_1)^i\cdot(r'_2-r_2+p^{\ell_2}x_2)^{e-i}\,. \quad (4.4)$$

The valuation of this expression is the minimum of $v_p(c_i(\mathbf{r}))+\ell_1 i+\ell_2(e-i)$, over all i. If this is $\geq k$ then $\mathbf{x}$ can take any value in $\mathbb{Z}/p^{k-\ell_1}\mathbb{Z}\times\mathbb{Z}/p^{k-\ell_2}\mathbb{Z}$, and extend, by the above equation, to a root mod p^k. This is what should happen as we are not in the linear-representative case. We need to count these possibilities, which will give us the number of ways the root $\mathbf{r}$ could lift as k increases. For this, we should consider only those possibilities of (ℓ_1,ℓ_2) that are *minimal*, i.e. (ℓ_1-1,ℓ_2) and (ℓ_1,ℓ_2-1) should violate the linear-inequality system. This is a system of half-spaces in the plane, forming an open polygon $\mathcal{P}$ with either $\leq e$ vertices or just one hyperplane. It can be checked that in all cases the counting function

$$\sum_{(\ell_1,\ell_2)\in\mathcal{P}} p^{(k-\ell_1)}\cdot p^{(k-\ell_2)} \quad (4.5)$$

can be rewritten as a sum of $p^{u_i(k)}$, $i\leq e$, where $u_i(k)$ is a linear function in k over $\mathbb{Q}$. Let us call this sum $N_{k,\mathbf{r}}(f)$. Importantly, the number of summands here is fixed, and does not grow with k.

The following example illustrates the notion of this polytope.

Example 4.11. *Consider the polynomial $f(x_1,x_2)=x_1^2x_2 \bmod p^k$. Here, for the root $(0,1)$, the value of e is 2, and accordingly ℓ_1, the precision of x_1 required, is $=k/2$. However, the value of e for $(1,0)$ is 1 and $\ell_2=k$. For the root $\mathbf{0}$, both variables contribute powers of p, where they are zero with precision ℓ_1,ℓ_2 respectively. Then, we must have $2\ell_1+\ell_2\geq k$, which gives the hyperplane in ℓ_1,ℓ_2; summing over all these values we can calculate Eqn.4.5.*

4.2.2 Linear-representative roots $\mathcal{L}$

Consider a linear-representative root $\mathbf{r} \in \mathcal{L}$, with fixed part as $\mathbf{a}$ of length (e_1, e_2) respectively. Up to linear transformations, we can claim that our algorithm defines a computable $\mathbb{Z}_p$-function $\mu(\cdot)$ s.t. for all $u \in \mathbb{Z}_p$, $f(x_1+a_1+p^{e_1}u\,, x_2+a_2+p^{e_2}\mu(u)) = 0$. Using this fact, we write Equation 4.3 in the ideal form, defining (e, v) as the largest integers s.t., in $\mathbb{Z}_p[\mathbf{x}]$,

$$f(x_1 + a_1\,, x_2 + a_2 + p^{e_2}\mu(0)) \;\in\; p^v \cdot \langle x_1\,, x_2\rangle^e \;+\; \langle x_1\,, x_2\rangle^{e+1}\,. \tag{4.6}$$

Note that we have defined (e, v) by fixing $u = 0$. The motivation is that if we use some other u in $\mathbb{Z}_p$, we will get the same values (e, v). Otherwise, say for some $0 \neq u \in \mathbb{Z}_p$,

$$f(x_1 + a_1 + p^{e_1}u\,, x_2 + a_2 + p^{e_2}\mu(u)) \;\in\; p^{v'} \cdot \langle x_1\,, x_2\rangle^{e'} \;+\; \langle x_1\,, x_2\rangle^{e'+1}\,.$$

Then, by Lemma 4.6, $e' = e$, because the tree remains isomorphic, even when we make the root more precise than $k \geq k_0$. In the same algorithm, lifting steps cause division by p-powers and reach the leaf $\mathbf{r}$, so v' remains v even when we make the root more precise than $k \geq k_0$.

Fix $u \in \mathbb{Z}/p^{k-e_1}\mathbb{Z}$, and consider the unique p-adic root $\mathbf{r}'$ that the leaf $\mathbf{r}$ gives above. We can now follow the counting process done after Equation 4.4. Varying u, the polytope boundary $\mathcal{P}$ does not change (similar to the argument given after Equation 4.6); on the other hand, fixing $(\ell_1 - e_1)$-digits of u (resp. ℓ_1 value), fixes that many in $\mu(u)$ (resp. ℓ_2 value). Thus, we get a *partial* count (slightly different from Equation 4.5) as:

$$\sum_{(\ell_1,\ell_2)\,\in\,\mathcal{P}\,\cup\,\{(k-e_2)-(\ell_1-e_1)\,\leq\,(k-\ell_2)\}} p^{(\ell_1-e_1)} \cdot p^{(k-e_1)-(\ell_1-e_1)} \cdot p^{(k-e_2)-(\ell_1-e_1)}\,. \tag{4.7}$$

We call this sum $N'_{k,\mathbf{r}}(f)$; and is written as a sum of $p^{u_i(k)}$, $i \leq e$, where $u_i(k)$ is a *linear* function in k over $\mathbb{Q}$.

4.2.3 Blowing-up root 0

What is missing here are the roots with valuation in the interval $[k_0, k-1]$, because these are zero mod p^{k_0} and so they get missed in $\mathcal{L}$ and $\mathcal{R}$. To account for these, we need to resort to the set $\mathcal{H}_{k_0}$, constructed in Theorem 4.10 that 'blows-up' the leaf node **0** in the tree of finding roots modulo p^{k_0}.

Now, from the way Algorithm 3 works, the representatives in $\mathcal{H}_{k_0}$ generate a disjoint set of $(\mathbb{Z}/p^k\mathbb{Z})$-roots, which are in number $= \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}$. This sum is easy to precompute (by Theorem 1.1), as it is independent of k. Each of these roots can be multiplied by p^e, for $e \in [k_0 \ldots k-1]$, to get the 'high'-valuation roots. Thus, the total number of such roots is $= (k-k_0) \cdot \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}$.

Overall, the above summands account for all the $\mathbb{Z}/p^k\mathbb{Z}$-roots of f, for $k \geq k_0$.

4.2.4 Computation of Igusa's LZF

The number of roots modulo p^k, for $k < k_0$, can be counted by Theorem 1.1. Fixing $k = k_0$, as described above, we compute the data related to the fixed tree; which has the linear-representative roots in $\mathcal{L}$, $\mathcal{H}_{k_0}$, and the remaining representative-roots in $\mathcal{R}$.

Then, the number of roots of f modulo p^k, $N_k(f)$, is given by

$$N_k(f) \;=\; \sum_{\mathbf{r}\in\mathcal{R}} N_{k,\mathbf{r}}(f) \;+\; \sum_{\mathbf{r}\in\mathcal{L}} N'_{k,\mathbf{r}}(f) \;+\; (k-k_0)\cdot \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}\,. \tag{4.8}$$

Recall that the number of roots modulo p^k due to any representative root $\mathbf{r}$ (or a leaf in the fixed tree in Theorem 4.10) is $N_{k,\mathbf{r}}(f)$ resp. $N'_{k,\mathbf{r}}(f)$; defined separately in Equations 4.5 and 4.7.

Now, the Poincaré series is given by (eg. see [DS20])

$$P(t) \;=\; P_0(t) + \sum_{\mathbf{r}\in\mathcal{R}} P_{\mathbf{r}}(t) + \sum_{\mathbf{r}\in\mathcal{L}} Q_{\mathbf{r}}(t) + \left(\sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}\right)\cdot \sum_{k\geq k_0}(k-k_0)\cdot(t/p)^k \;, \quad (4.9)$$

where $P_0(t) = \sum_{k<k_0} N_k(f)\cdot(t/p)^k$, $P_{\mathbf{r}}(t) := \sum_{k\geq k_0} N_{k,\mathbf{r}}(f)\cdot(t/p)^k$ and $Q_{\mathbf{r}}(t) := \sum_{k\geq k_0} N'_{k,\mathbf{r}}(f)\cdot(t/p)^k$.

The expression of Equation 4.8 is a sum of either $p^{u_i(k)}$ or $u_i(k)$, $i \leq O(d)$, where $u_i(k)$ is a *linear* function in k over $\mathbb{Q}$. Thus, Equation 4.9 can be easily expressed in a 'closed-form formula' by summing the geometric progression over k's, as was done in [DS20, Lemma 23]. Thus, $P_{\mathbf{r}}(t), Q_{\mathbf{r}}(t)$ are rational functions in $\mathbb{Q}(t)$. Therefore, the rational function for the Poincaré series $P(t)$ is computable as promised in Corollary 4.2.

Using this technique for finding all the roots modulo p^k for n-variates, we can generalize the algorithms for finding $\mathbb{Z}_p$-roots and computing the Igusa local zeta function for a system of m polynomials in n-variables as well. For finding roots in $\mathbb{Z}_p$, we consider the resultant w.r.t. one variable at a time, find a bound similar to Lemma 4.4, and proceed with the analysis of roots which are in $\mathbb{Z}_p$ or are not roots of the discriminant. At each step, the bounds according to one variable at a time will be obtained, which get multiplied to give a bound k_0 for $f(\mathbf{x})$ such that roots of $f(\mathbf{x})$ mod p^{k_0} gives us roots which correspond to $\mathbb{Z}_p$ roots as well. This will be a generalization of Theorem 4.10 to n-variates. Similarly, we can count roots, where the number of possibilities due to linear-representative roots depends on the rank of the linear forms, and the sum will again be a rational form.

The complexity of finding $\mathbb{Z}_p$ points and that of computing Igusa local zeta function will remain deterministic $\mathsf{poly}((m\log M + p + d)^{(2d(n-1))^{n-1}})$-time.

Chapter 5

Solvability of system of polynomial equations over Galois rings

In this chapter, we generalize the result of Huang and Wong [HW99], to give an algorithm to solve Hilbert's Nullstellensatz over Galois rings. We prove Theorem 1.2 by giving an algorithm to compute a common solution to the system of polynomial equations.

Theorem 1.2 (HN_{p^k})**.** *Let $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{Z}[z][\mathbf{x}]$ be a set of n-variate degree d polynomials, where n is a constant. Given a prime power p^k for constant k and an $\mathbb{F}_p$-irreducible polynomial $\varphi(z)$, we can efficiently find a common root of the system $f_i(\mathbf{x}) \equiv 0 \bmod \langle \varphi(z), p^k \rangle$, for $i \in [m]$, in randomized $\mathsf{poly}(d^{c_{nk}}, m, \deg(\varphi), \log p)$ time.*

5.1 Overview of the algorithm

In this chapter, we describe a method to find roots of a system of polynomials over a Galois ring.

We prove Theorem 1.2 by giving an algorithm that returns False if the given system of n-variate polynomials $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{Z}[z][\mathbf{x}]$ has no root in Galois ring $\mathbb{G} := \mathbb{Z}[z]/\langle p^k, \varphi(z) \rangle$, otherwise outputs a possible root. This is the problem of solving Hilbert's Nullstellensatz over the Galois ring $\mathbb{G}$. The idea, as hinted before, is to first reduce this problem to the base field $\mathbb{G}/\langle p \rangle = \mathbb{Z}[z]/\langle p, \varphi(z) \rangle \cong \mathbb{F}_p[z]/\langle \varphi(z) \rangle$.

Since $\varphi(z)$ is irreducible over $\mathbb{F}_p$ of degree b, $\mathbb{F}_p[z]/\langle\varphi(z)\rangle$ is also a field which is a finite extension of $\mathbb{F}_p$, denoted as $\mathbb{F}_q$ for $q := p^b$. Now, since finite fields of given size are unique up to isomorphisms [Moo93, Sec.3], we restrict ourselves to the construction of $\mathbb{F}_q$ given by $\mathbb{F}_p[z]/\langle\varphi(z)\rangle$. Throughout this algorithm, we will use [HW99] to solve Hilbert's Nullstellensatz over finite field $\mathbb{F}_q$, which requires an additional condition that q must be large enough, i.e. $q > d^{(nk)^{\Omega((nk)^2)}}$. If q is smaller, then anyways the brute-force search, for $\mathbb{G}$-roots, can be done in time q^{nk}.

5.1.1 Breaking into digits

The method of finding roots is performed iteratively on each p-adic coordinate (or *digit*). We find (virtual) roots at each step, and using these roots, find those corresponding to higher coordinates. The method has been employed in univariates [BLQ13; DMS19; DMS21; NRS17]. For a root $\mathbf{a} \in \mathbb{F}_q^n$ (also embedded in $\mathbb{G}^n$) of the system of polynomials modulo p, we transform each of the polynomials $f_j(\mathbf{x})$ to $f_j(\mathbf{a} + p\mathbf{x})$ for $j \in [m]$, inspired by the p-adic coordinates. Since we have standard methods to find roots in $\mathbb{F}_q$, while the same is difficult in Galois rings, we divide-out the 'excess' powers of p, to bring this system back to $\mathbb{F}_q$. These excess powers of p will be given by $v_j = v(f_j(\mathbf{a} + p\mathbf{x}))$, which will be termed as *val-multiplicity* of the root $\mathbf{a}$. The step thus discussed, given by transforming the polynomial $f_j(\mathbf{x})$ to $p^{-1}f_j(\mathbf{a} + p\mathbf{x})$ will be called the *lifting step*. Point $\mathbf{a}$ will be termed as the *local root* at that lifting step. We could try a 'faster' lift, $p^{-v_j}f_j(\mathbf{a} + p\mathbf{x})$; however this idea fails, if $v_j \geq 2$, due to some intermediate mod p arithmetic that our Algorithm uses.

The modification to the polynomial during lifting will make sure that the $\mathbb{F}_q$ coordinates at the t-th step of lifting will return the t-th p-adic coordinate. For example, if $\mathbf{a}$ is an $\mathbb{F}_q$-root of $f_j(\mathbf{x})$, and after lifting, the polynomial becomes $\tilde{f}_j(\mathbf{x}) := p^{-1}f_j(\mathbf{a} + p\mathbf{x})$ which has an $\mathbb{F}_q$-root $\mathbf{b}$, then $(a_1 + pb_1, \ldots, a_n + pb_n)$ is a root of $f_j(\mathbf{x})$ mod p^2. Note that some local roots might not have liftings while others can; as illustrated by the following example.

Example 5.1. *Consider* $f(x_1, x_2) := x_1^3 - x_2^3 + 5$ *and* $p := 5$. $(0, 0)$ *and* $(1, 1)$ *are its*

$\mathbb{F}_p$-roots. When we start the root $(0,0)$, the lifting step given by the transformation $(x_1, x_2) \mapsto (5x_1, 5x_2)$ and subsequent division by 5, yields the polynomial $25x_1^3 - 25x_2^3 + 1$ which does not have $\mathbb{F}_5$-roots. Although, restarting with the root as $(1,1)$ yields the polynomial $25x_1^3 - 25x_2^3 + 15x_1^2 - 15x_2^2 + 3x_1 - 3x_2 + 1$ after lifting, which now has $(3,0)$ as its $\mathbb{F}_5$-root! This anomaly is explained by a curious fact: $(0,0)$ is a singular root of f, while $(1,1)$ is non-*singular.*

5.1.2 Virtual roots

The algorithm for univariate root finding [BLQ13] implicitly enumerates over all possible $\mathbb{F}_q$-roots, to check which one lifts. It is not possible for us to enumerate over all roots as already for curves, $\Omega(p)$ roots might exist, and there is no standard way of representing them 'compactly'. To tackle this problem, we introduce 'formal' variables $(y_{i,1}, \ldots, y_{i,n})$ for the roots corresponding to the i-th lifting step, instead of fixing them to $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$, and lift to the next step to form a new polynomial $\tilde{f}$ in terms of this $\mathbf{y}_i$. We will denote this tuple $(y_{i,1}, \ldots, y_{i,n})$ as *virtual root*. At each step, we need the property of $\mathbf{y}_i$ that it is a root modulo p. In order to track these properties together, we create a (p-adic) ideal $\hat{\mathtt{I}}$; which can be thought of as a 'data structure' that stores all possible roots, but in 'higher'-precision p-adics. At each step, we include the polynomials $f_j(\mathbf{y}_i) \bmod p$ to this ideal, for $j \in [m]$; factorize, and lift again to p-adics. E.g. this ideal vanishing at a point $\mathbf{a} \in (\mathbb{Z}[z]/\langle\varphi(z)\rangle)^n$ implies: the virtual root $\mathbf{y}_i$ can be realized as an 'actual' root $\mathbf{a}$ of $f_j(\mathbf{x})$. A similar idea of storing roots via ideals, though in a simpler setting, has been employed for univariate polynomials in [Che+19; DMS19; DS20].

5.1.3 Lifting mod irreducible components

As discussed before, we develop the idea of formation of p-adic ideals, which will store the roots of each step of lifting. In the first step, we have the system of (p-adic) polynomials $f_j(\mathbf{x})$, $j \in [m]$. If this system has a local root $\mathbf{a}$, then we perform lifting to get the polynomials $\tilde{f}_j(\mathbf{x}) := p^{-1} f_j(\mathbf{a} + p\mathbf{x})$ and move on to find the roots of $\tilde{f}_j$.

Here, we use formal variables, for local roots and then perform lifting, instead of fixing them as field-values. We first consider the virtual local root as $(y_{0,1}, \ldots, y_{0,n})$ and add the polynomials $f_j(\mathbf{y}_0) \bmod p$, $j \in [m]$, to the ideal $\mathtt{I}$. We will use $\hat{\mathtt{I}}$ for the $(\mathbb{Z}_p[z]/\langle\varphi(z)\rangle)$-ideal, which is a p-adic lifting of $\mathtt{I}$ to $\mathbb{Z}_p$. This lifting of the ideal from $\mathbb{F}_q$ to the *unramified p-adic integer* ring $\widehat{\mathbb{G}} := \mathbb{Z}_p[z]/\langle\varphi(z)\rangle$ will be explained in more details later. Due to this motivation of lifting, instead of using $\hat{\mathtt{I}}$, we use an *irreducible component* of $\hat{\mathtt{I}} + \langle p\rangle$ lifted to p-adics, denoted as $\hat{\mathtt{C}}$, as we will see later on. For the lifting step, we will have the polynomial after lifting, denoted by $\tilde{f}_j(\mathbf{x}) := p^{-1} f_j(\mathbf{y}_0 + p\mathbf{x}) \bmod \hat{\mathtt{C}}$; where the ideal arithmetic is over $\widehat{\mathbb{G}}$.

5.1.4 Growing the p-adic ideal

In this process of forming next-precision polynomials and ideals, we use $\widehat{\mathbb{G}}$-arithmetic, instead of $\mathbb{F}_q$ in the base, as it handles division by p in a clean way. Let us assume that we have lifted the system of polynomials $f_j(\mathbf{x})$, $j \in [m]$, for ℓ-steps, to give the polynomials $\tilde{f}_j(\mathbf{x}) \in \widehat{\mathbb{G}}[\mathbf{y}_0, \ldots, \mathbf{y}_{\ell-1}][\mathbf{x}]$, and the ideal $\hat{\mathtt{I}} \subseteq \widehat{\mathbb{G}}[\mathbf{y}_0, \ldots, \mathbf{y}_{\ell-1}]$.

Next, we consider the virtual local root $\mathbf{y}_\ell =: (y_{\ell,1}, \ldots, y_{\ell,n})$ of the system $\tilde{f}_j(\mathbf{x})$, $j \in [m]$, and perform lifting to the $(\ell+1)$-th precision, while we store information about the ℓ-th step by adding the polynomials $f_j(\mathbf{y}_\ell)$ into $\mathtt{I}$, for $j \in [m]$, and redefine $\hat{\mathtt{I}}$; thereby growing the precision of the roots contained inside $\hat{\mathtt{I}}$. However, followed by this, we perform decomposition of this ideal into *absolutely* irreducible components, on which we perform our arithmetic, to redefine $\tilde{f}_j$. In case this gives new constraints on previous variables, we *backtrack* the steps.

5.1.5 Finding a satisfying instance

After iteratively forming a chain of ideals while increasing the precision of the roots, we check if the system has a solution. The root of the polynomial which was, say, present at the beginning of the algorithm will be of the form

$$(a_{0,1} + a_{1,1}p + \cdots + a_{k-1,1}p^{k-1}, \ldots, a_{0,n} + a_{1,n}p + \cdots + a_{k-1,n}p^{k-1}). \tag{5.1}$$

Here, j-th coordinate $a_{i,j}$ is in $\widehat{\mathbb{G}}$. So, this expansion is *not* unique (eg. we can subtract any number t from $a_{1,1}$ and add $p \cdot t$ to $a_{0,1}$). Also, it suffices, for our application, to find values $a_{i,j}$ of the virtual root $y_{i,j}$ up to the precision of $\mathbb{G}$. We achieve this by:

5.1.6 Finding non-singular roots

After setting the virtual roots in order to achieve the required prime power k, say that we have the ideal $\hat{\mathtt{I}}$, birationally mapped to a hypersurface $\hat{\mathtt{H}}$, given by a polynomial $\hat{h}$ over $\widehat{\mathbb{G}}$, of dim $= r$, for $0 \leq r \leq \dim(\mathtt{I})$. While, $\mathtt{I}$ $(= \hat{\mathtt{I}} + \langle p \rangle)$ is birationally mapped to a hypersurface $\mathtt{H}$, given by a polynomial h over $\mathbb{F}_q$. We find a random root of $\mathtt{H}$ [HW99, Thm.2.6] and map it to the 0-th coordinate of roots of $\hat{\mathtt{H}}$; this crucial property is proved in Lemma 5.3. After which we can lift to find a $\widehat{\mathbb{G}}$-root using an easy variant of Hensel's lifting (Theorem 3.2-(2)). This procedure gives us a $\widehat{\mathbb{G}}$-root from the $\mathbb{F}_q$-root.

Now, the density of non-singular roots on $\mathtt{H}$ will be much greater than singular roots (Lemma 5.7) if it is *absolutely irreducible*. Picking a non-singular $\mathbb{F}_q$-root, at random, we can lift it to a $\mathbb{G}$-root. If $\mathtt{H}$ is *relatively irreducible* (i.e. reduces in some field extension), we add the first-order derivative of h to the ideal (Lemma 5.2), as will be explained in Section 5.2. Thus, we lift a root whenever the hypersurface is absolutely irreducible and satisfiable.

5.1.7 Branching out by absolutely irreducible components

Our objective is to obtain non-singular roots, via the birational hypersurface, as they lift all the way to p-adics. So, we 'replace' the ideal $\mathtt{I}$ either by its absolutely irreducible components. In the latter case, since the ideal is radical, each root lifts to $\widehat{\mathbb{G}}$ too (Lemma 5.8).

Thus, as discussed in the above paragraph (on finding non-singular roots), we decompose the ideal $\mathtt{I} = \hat{\mathtt{I}} + \langle p \rangle$, the projection of $\hat{\mathtt{I}}$ to $\mathbb{F}_q$, using the decomposition algorithm [HW99, Sec.3.3], and then lift them again to $\widehat{\mathbb{G}}$ using Lemma 5.3. The

decomposition procedure may give 'many' absolutely irreducible components over $\mathbb{F}_q$, on each of which our $\mathrm{HN}_{p^k}()$ algorithm recurses. This procedure— of selecting an irreducible component $\mathtt{C}$ each time, growing that ideal to next precision, and again loop over its irreducible components —can be seen as a *tree* $\mathcal{T}$.

This tree has several nodes which correspond to absolutely irreducible ideals. The depth of the tree represents the precision of the roots formed until that point. This is also the number of times the polynomials have been lifted, and thus the ideal has grown. The tree gives branches which correspond to nodes containing irreducible components of the ideal formed from growing the parent ideal to the next precision coordinate. We will show in Theorem 5.4 that these ideals can be used to recover a $\mathbb{G}$-root of the system of polynomials, and Algorithm 5 returns the leaves $\mathcal{L}$, with the tree $\mathcal{T}$, thus formed.

5.1.8 Main algorithm

Using these ideas, and some technicalities on decomposition into absolutely irreducible components described in Section 5.2, we sketch our algorithm, which has been developed in Dwivedi's thesis [Dwi22].

Input: Given the system of polynomials, it returns all the leaves of the tree described in the previous paragraph. The input is given as m n-variate polynomials $\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \mid f_j(\mathbf{x}) \in \widehat{\mathbb{G}}[\mathbf{y}_0, \ldots \mathbf{y}_{\ell-1}][\mathbf{x}]\}$ each of degree at most d, the exponent k, an ideal $\hat{\mathtt{I}} = \hat{\mathtt{I}}_{\ell-1} \subseteq \widehat{\mathbb{G}}[\mathbf{y}_0, \ldots \mathbf{y}_{\ell-1}]$. These ideals are given as a tree $\mathcal{T}$ datastructure, as discussed in Section 5.1.7.

Output: The algorithm outputs a list $\mathcal{L}$ of (absolutely irreducible) ideals, which contains the lift of the common roots of the system $f_j(\mathbf{x}) \equiv 0 \bmod \langle p^k \rangle + \hat{\mathtt{I}}$, for $j \in [m]$. It also returns a tree data-structure which gives the path to each of these ideals from the node.

Initialization: We initialize the ideal as $\hat{\mathtt{I}} := \langle 0 \rangle$, $\ell := 0$, and the required exponent as k. A system of polynomials $\mathcal{F} := \{F_1(\mathbf{x}), \ldots, F_m(\mathbf{x})\}$ where $F_j(\mathbf{x}) \in \widehat{\mathbb{G}}[\mathbf{x}]$. We pass $\mathcal{F}$ to the algorithm so it starts with $\mathrm{HN}_{p^k}(F_1, \ldots, F_m, k, \langle 0 \rangle)$.

Algorithm 5 Algorithm to find roots of a system of polynomial equations over a Galois ring [Dwi22].

procedure $\mathrm{HN}_{p^k}(f_1, \ldots, f_m, k, \hat{\mathtt{I}})$

  if Zeroset $\mathbf{V}_{\mathbb{F}_q}(\hat{\mathtt{I}} + \langle p \rangle) = \emptyset$ **then return** $\{\}$.

  if $k \leq 0$ **then return** $\{\hat{I}\}$.

  $\mathtt{I} \leftarrow \langle f_1(\mathbf{y}_\ell), \ldots, f_m(\mathbf{y}_\ell) \rangle + \hat{\mathtt{I}} + \langle p \rangle$, for (new) virtual root $\mathbf{y}_\ell := (y_{\ell,1}, \ldots, y_{\ell,n})$.

  $\mathcal{S} \leftarrow$ Abs_Decomp($\mathtt{I}$); absolutely irreducible ideals as computed by Algorithm 6.

  $\mathcal{L} \leftarrow \{\}$

  for each $\mathtt{C} \in \mathcal{S}$ **do**

    if $\mathtt{C} \cap \mathbb{F}_q[\mathbf{y}_0, \ldots, \mathbf{y}_{\ell-1}] = \mathtt{I} \cap \mathbb{F}_q[\mathbf{y}_0, \ldots, \mathbf{y}_{\ell-1}]$ **then**

      Find the special lift $\hat{\mathtt{C}}$ of $\mathtt{C}$ to $\hat{\mathbb{G}}$ by computing Gröbner basis and lifting, using Lemma 5.3. /*$\hat{\mathtt{C}}$ is *prime*; reduced Gröbner basis w.r.t. $\mathbf{y}_0 < \ldots < \mathbf{y}_{k-1}$.*/

      For $j \in [m]$, compute $\tilde{f}_j(\mathbf{x}) := p^{-1} f_j(\mathbf{y}_\ell + p\mathbf{x}) \bmod \hat{\mathtt{C}}$, over $\widehat{\mathbb{G}}[\mathbf{y}_0, \ldots \mathbf{y}_\ell][\mathbf{x}]$.

      $\mathcal{L} \leftarrow \mathcal{L} \cup \mathrm{HN}_{p^k}(\tilde{f}_1, \ldots, \tilde{f}_m, k-1, \hat{\mathtt{C}})$. /*Maintain the recursion-tree $\mathcal{T}$.*/

    else /*Backtrack & repeat steps*/

      Find min $s \leq \ell - 1$ s.t. $\mathtt{C} \leftarrow \mathtt{C} \cap \mathbb{F}_q[\mathbf{y}_0, \ldots, \mathbf{y}_s] \supsetneq \mathtt{I} \cap \mathbb{F}_q[\mathbf{y}_0, \ldots, \mathbf{y}_s]$.

      Find special lift $\hat{\mathtt{C}}$ of $\mathtt{C}$ over $\hat{\mathbb{G}}$ using Lemma 5.3.

      For all $j \in [m]$, compute $\tilde{f}_j(\mathbf{x}) := p^{-s-1} F_j(\mathbf{y_0} + \cdots + p^s \mathbf{y}_s + p^{s+1}\mathbf{x}) \bmod \hat{\mathtt{C}}$.

      $\mathcal{L} \leftarrow \mathcal{L} \cup \mathrm{HN}_{p^k}(\tilde{f}_1, \ldots, \tilde{f}_m, k+\ell-1-s, \hat{\mathtt{C}})$. /*Maintain $\mathcal{T}$ as before.*/

  return $\mathcal{L}$. /*Also, return the recursion-tree $\mathcal{T}$ whose leaves are ideals in $\mathcal{L}$.*/

Simple invariant. A node in the recursion-tree $\mathcal{T}$ either moves from $\hat{\mathtt{I}}_{\ell-1}$ to $\hat{\mathtt{I}}_\ell$, or *backtracks* to redefine $\hat{\mathtt{I}}_s$, $s < \ell$. In the former case, k reduces, while in the latter case $\dim(\mathbf{V}(\hat{\mathtt{I}}_s))$ reduces. Thus, in a path, $\hat{\mathtt{I}}_s$ can be redefined at most n times; thus

bounding the length of any path in the tree by $\leq k + kn$.

5.2 Mapping $\mathbb{F}_p$ roots to $\mathbb{Z}_p$ roots

As we have seen in the previous section, we need to work with ideals over p-adics, $\hat{\mathbb{G}}$, since we need to allow division by p. In this section, we establish a connection between the ideals over $\mathbb{F}_q$ and those over $\hat{\mathbb{G}}$ such that from the roots stored in $\mathtt{I}$ over $\mathbb{F}_q$, we can find the corresponding $\hat{\mathbb{G}}$-roots of $\hat{\mathtt{I}}$.

In order to find this mapping, we use the decompose the ideals into absolutely irreducible components. Since we can find non-singular roots w.h.p. when the ideal is absolutely irreducible, the val-multiplicity will be 1 (Corollary 3.5). Now, using Theorem 3.2 Part (2), we can lift the root to p-adics.

Now, in order to find the absolutely irreducible components of the ideal, we modify the machinery of the decomposition algorithm developed in [HW99, Sec.3]. The crux of the algorithm is the following lemma, which will be used to give an invariant on the decomposition algorithm.

Lemma 5.2 (Singular roots [HW99, Lem.2.1])**.** *If a polynomial $h(\mathbf{x})$ is irreducible over $\mathbb{F}_q$, but reducible over its algebraic closure $\overline{\mathbb{F}}_q$, then for any root $\mathbf{a} \in \mathbb{F}_q^n$ of h, we have*

$$h(\mathbf{a}) = h_{x_j}(\mathbf{a}) = 0$$

over $\mathbb{F}_q$, for any first-order partial-derivative h_{x_j} of h.

5.2.1 Decomposition via birationally equivalent hypersurfaces

First, given the ideal $\mathtt{C}$ irreducible over $\mathbb{F}_q$, we construct a birationally equivalent hypersurface and lift an $\mathbb{F}_q$-root to $\hat{\mathbb{G}}$-root using Theorem 3.2. The connection between these $\mathbb{F}_q$ roots and $\hat{\mathbb{G}}$ roots has been shown in Lemma 5.3.

We mainly follow the techniques of [HW99, Sec.3.3], by mapping each $\mathbb{F}_q$ ideal $\mathtt{C}$

to a union of (finitely many) irreducible components of C. We can map each of these irreducible ideals to a hypersurface $\mathsf{H} = \mathbf{V}(h)$. If h is not absolutely irreducible, then by Lemma 5.2, we can keep adding the pullback of a non-zero derivative h' of h, given by h^*, and further decomposing the ideal thus formed. This procedure reduces the dimension of the ideal C; thus, it continues for only few steps, and reaches absolutely irreducible components.

5.2.2 Loss of points

The map between a component ideal and its corresponding birationally equivalent hypersurface is given as rational functions $\psi_2 : \mathbf{V}(\mathsf{C}) \to \mathsf{H}$ and $\psi_1 : \mathsf{H} \to \mathbf{V}(\mathsf{C})$.

Due to this property, some C-roots might get lost due to the image of ψ_2 being zeroes of the denominator of the rational function ψ_1 (since ψ_2 is linear, like the construction idea of primitive element theorem, and no roots will be lost). In order to include these points, we consider the polynomial e, which is the product of the denominators of ψ_1. However, since e is a polynomial over the base field of H, we bring it back to that of C by considering its pullback e^*. This avoids the loss of any $\mathbb{F}_q$-point when we decompose into absolutely irreducible hypersurfaces.

5.2.3 Special lift of ideals and roots.

We are forming the ideals by adding $f_j(\mathbf{y}_\ell)$ mod p into $\mathtt{I}$ and then 'lifting' the ideal to the p-adic one over $\widehat{\mathbb{G}}$. Which $\mathbb{F}_q$-roots lift 'smoothly' to p-adics, and which don't? The latter ones are handled separately (as discussed above).

Lemma 5.3 (Connection of points via hypersurfaces)**.** *Given an $\mathbb{F}_q$-irreducible ideal C (resp. its birational equivalent hypersurface H), we can lift it to a* prime *$\widehat{\mathbb{G}}$-ideal $\hat{\mathsf{C}}$ (resp. its birational equivalent hypersurface $\hat{\mathsf{H}}$), such that their morphism diagram commutes (Figure 5.1).*

In particular, for a non-singular $\mathbb{F}_q$-root of H (thus a root of C), we can find a $\widehat{\mathbb{G}}$-root of $\hat{\mathsf{H}}$; which gives a root of $\hat{\mathsf{C}}$. This sets up the 'connection' between roots of C and $\hat{\mathsf{C}}$.

Proof. We have a prime ideal C given by generators in $\mathbb{F}_q[y_1, \ldots, y_N]$. Let $r > 0$ be the dimension of the variety of C. By one of the definitions of dimension, there is a subset $B =: \{\ell_1 < \ldots < \ell_r\}$ of *least* possible variables in $\mathbf{y}$, such that the function field $\mathbb{F}_q(\mathsf{C})$ is a *finite* extension over the transcendental field $\mathbb{F}_q(B)$. So, we consider its defining maximal ideal $B^{-1}\mathsf{C}$; and compute its reduced Gröbner basis (using Buchberger's algorithm [Buc65]); with the graded lexicographical ordering $y_1 < \ldots < y_N$ and variables B *localized*. Let $B' := \mathbf{y} \setminus B =: \{\ell_{r+1} < \ldots < \ell_N\}$ be the remaining variables.

Triangular form. The localization $B^{-1}\mathsf{C}$ is a zero-dimensional prime ideal (= *maximal* ideal). Thus, by [GTZ88, Prop.5.9], $B^{-1}\mathsf{C}$ has exactly $N - r$ generators, the i-th one ($r < i \leq N$) corresponding to a monic *minpoly* (over $\mathbb{F}_q(B)$) for the variable ℓ_i in B' (in particular, having the leading-monomial an ℓ_i-power). Thus, the Gröbner basis $\mathrm{GB}(B^{-1}\mathsf{C})$ is in a special form, that we call the *triangular form* in B' over B (see [DMS19, Def.4]).

p-adic lift. Compute the reduced Gröbner basis $\mathrm{GB}(\mathsf{C})$ too, and divide each generator by its leading coefficient (in $\mathbb{F}_q^*$) to make the polynomials *monic*; store them in reduced form where the coefficients are in $\{0, \ldots, p-1\}$. Define the p-adic lift $\hat{\mathsf{C}}$, of C to $\hat{\mathbb{G}}$, by considering the trivial integral embedding of each generator of C. By Gröbner basis properties and the special generators, this special lift $\hat{\mathsf{C}}$ is a prime $\hat{\mathbb{G}}$-ideal.

Doing the same thing to $\mathrm{GB}(B^{-1}\mathsf{C})$, it is easy to deduce: the $\hat{\mathbb{G}}$-ideal thus obtained, called $B^{-1}\hat{\mathsf{C}}$, is a maximal ideal with a triangular (& reduced) Gröbner basis.

$\mathbb{F}_q$-map. By construction, $\mathbb{F}_q(B)[B']/\mathsf{C}$ is a field, denoted R, of finite degree over $\mathsf{R}_0 := \mathbb{F}_q(B)$. We can compute a hypersurface H that is *birationally equivalent* to the variety of C [HW99, Thm.2.6]. A standard algebraic way to compute it, is to pick a *random* linear form ℓ_0; assume q to be 'large' enough fo random sampling. Let $h(Y)$ be the *minpoly* of the *primitive* element $\ell_0 \in \mathsf{R}$ over the subfield R_0. We can store a representation of h in $\mathbb{F}_q[B][Y]$ such that it gives an R_0-isomorphism ψ_1 between

the fields, $\mathsf{R} = \mathsf{R}_0[B']/\mathsf{C} \cong \mathsf{S} := \mathsf{R}_0[Y]/\langle h \rangle$; mapping $\ell_0 \mapsto Y$, and other ℓ_i $(i > r)$ to its implied image.

***p*-adic map.** Take any p-adic lift $\hat{h}$ of h; clearly $\hat{h} \in \widehat{\mathbb{G}}(B)[Y]$. By definition, $\hat{h}(\ell_0) \in \mathsf{C} = \hat{\mathsf{C}} + \langle p \rangle$. Since ℓ_0 is a separable $\mathbb{F}_q$-root of $\hat{h}$, we can Hensel lift it to a $\widehat{\mathbb{G}}$-root $\ell_0' \in \widehat{\mathbb{G}}(B)[B'] =: \mathsf{R}_0'[B']$ such that $\hat{h}(\ell_0') \in \hat{\mathsf{C}}$. So, mapping $Y \mapsto \ell_0'$ gives a R_0'-homomorphism $\hat{\psi}_2 : \mathsf{S}' := \mathsf{R}_0'[Y]/\langle \hat{h} \rangle \longrightarrow \mathsf{R}' = \mathsf{R}_0'[B']/\hat{\mathsf{C}}$; which is a map between integral domains. Moreover, it remains a nontrivial homomorphism if we localize the base ring from $\mathbb{Z}_p$ to $\mathbb{Q}_p$; making it a map between *fields*. Thus, $\hat{\psi}_2$ is an injective R_0'-homomorphism.

Now we know: all the four rings in Figure 5.1 are domains (& two are fields). So, in case $\hat{\psi}_2$ is not an isomorphism, it is injective and non-surjective. Let $v_0 \in \mathsf{R}'$ be an element that is out of the image, but we know that some lift $v_0 + pv_1$ is in the image of $\hat{\psi}_2$ (by traversing the commutative diagram). Similarly, we have that some lift $v_1 + pv_2$, of v_1, is in the image of $\hat{\psi}_2$. Combining these two, we know: $v_0 - p^2 v_2$ is in the image of $\hat{\psi}_2$. Doing this *ad infinitum*, we get v_0 in the image of $\hat{\psi}_2$; contradicting its choice. We conclude: $\hat{\psi}_2$ is an isomorphism, with the inverse map being (say) $\hat{\psi}_1$.

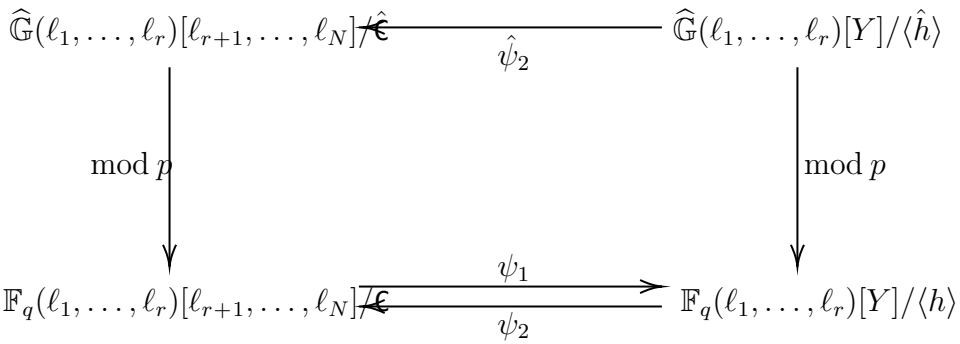


Figure 5.1: Commutative Diagram

In the above diagram let us start with a non-singular $\mathbb{F}_q$-root $\mathbf{a}$ of $\mathsf{H} := \mathbf{V}(h)$. With high probability, it will keep the relevant polynomials in $\ell_1, \ldots, \ell_r$ nonzero mod p; thus it would be consistent with the localization. It has 'pullback' via ψ_1, giving a root of C. By the separability of the $\mathbb{F}_q$-root, $\mathbf{a}$ lifts to a root $\hat{\mathbf{a}}$ of $\hat{\mathsf{H}} := \mathbf{V}(\hat{h})$; from up there it has 'pullback' via $\hat{\psi}_1$, giving a $\widehat{\mathbb{G}}$-root of $\hat{\mathsf{C}}$ too. This connects $\mathbf{V}(\mathsf{C})$ with $\mathbf{V}(\hat{\mathsf{C}})$. □

The proof of Lemma 5.3 however does not work for a 'small' set of points given by the roots of e since they do not have maps to $\mathtt{C}$. So, we separately include these points, by including the pullback e^*; and continue with our decomposition algorithm. Now, the root-lifting technique of Lemma 5.3 uses Hensel's lifting to lift non-singular points to p-adics. This fails when the root has val-multiplicity ≥ 1, for which we again need to add the pullback of h', say h^*, into the ideal, and continue with our decomposition Algorithm 6.

In the end, we have absolutely irreducible ideals, which lift over $\widehat{\mathbb{G}}$. Note that sometimes the absolutely irreducible ideals might be single points as well, of the form $\langle \mathbf{y} - \mathbf{a} \rangle$. When the ideals are absolutely irreducible, it is easy to search for a $\widehat{\mathbb{G}}$-root (see Section 5.3). Based on these ideas, we give Algorithm 6 to decompose into absolutely irreducible components, without losing any $\mathbb{G}$-root. It decomposes the $\mathbb{F}_q$-ideal $\mathtt{I}$ to absolutely irreducible ideals $\mathtt{C} \in \mathcal{S}_{abs}$ in such a way that zeroset $\mathbf{V}_{\mathbb{F}_q}(\mathtt{I})$ remains unchanged, i.e, $\mathbf{V}_{\mathbb{F}_q}(\mathtt{I}) = \bigcup_{\mathtt{C} \in \mathcal{S}_{abs}} \mathbf{V}_{\mathbb{F}_q}(\mathtt{C})$. This modification from [HW99] has been explained in [Dwi22] in more details.

Input: The algorithm takes as input, a radical ideal $\mathtt{I} \subseteq \mathbb{F}_q[y_1, \ldots, y_n]$.

Output: The algorithm outputs a set $\mathcal{S}_{abs}$ consisting of absolutely irreducible ideals $\mathtt{C}$, s.t. $\mathbf{V}(\mathtt{I}) = \bigcup_{\mathtt{C} \in \mathcal{S}_{abs}} \mathbf{V}(\mathtt{C})$.

Algorithm 6 Decomposing $\mathtt{I}$ into absolutely irreducible components over $\mathbb{F}_q$ [Dwi22].

1: **procedure** ABS_DECOMP($\mathtt{I}$)
2: Define $\mathcal{S}_{abs} := \{\}$ and $\mathcal{S}_{irr} := \{\}$.
3: Decompose $\mathtt{I}$ into irreducible components over $\mathbb{F}_q$ using [HW99], and store them in $\mathcal{S}_{irr}$.
4: **while** $\mathcal{S}_{irr} \neq \emptyset$ **do**
5: $\mathtt{C} \leftarrow \mathtt{Pop}(\mathcal{S}_{irr})$.
6: **if** $\dim(\mathbf{V}_{\mathbb{F}_q}(\mathtt{C})) = 0$ **then**
7: Compute $\mathbf{V}_{\mathbb{F}_q}(\mathtt{C})$ using [HW99] and for each $\mathbf{a} \in \mathbf{V}_{\mathbb{F}_q}(\mathtt{C})$, update $\mathcal{S}_{abs} \leftarrow \mathcal{S}_{abs} \cup \{\langle \mathbf{y} - \mathbf{a} \rangle\}$.
8: **else**

9: **if** $\mathtt{C}$ is absolutely irreducible **then**

10: $\mathcal{S}_{abs} \leftarrow \mathcal{S}_{abs} \cup \{\mathtt{C}\}$

11: Let $\dim(\mathbf{V}_{\mathbb{F}_q}(\mathtt{C})) =: r$. Using [HW99] compute a birationally equivalent hypersurface $\mathtt{H} := \mathbf{V}_{\mathbb{F}_q}(h(l_1, \ldots, l_r, Y))$ and the rational maps $\psi_1 : \mathtt{H} \to \mathbf{V}(\mathtt{C})$ and $\psi_2 : \mathbf{V}(\mathtt{C}) \to \mathtt{H}$. ($\mathbf{l}, Y$ are linear forms in $\mathbf{y}$; also see Figure 5.1.)

12: Compute $\mathtt{C}_1 := \mathsf{Rad}(\mathtt{C} + \langle h^* \rangle)$, where h^* is pullback of a first-order partial-derivative $h' \neq 0$.

13: Compute $\mathtt{C}_2 := \mathsf{Rad}(\mathtt{C} + \langle e^* \rangle)$, where e^* is the pullback of e, which is a product of the denominators that appear– in rational functions $\psi_1 =: (\psi_{1,1}, \ldots, \psi_{1,n})$, or in the localization done in Lemma 5.3.

14: Decompose the ideals $\mathtt{C}_1, \mathtt{C}_2$ into irreducible components over $\mathbb{F}_q$ using [HW99], and push these components into $\mathcal{S}_{irr}$.

15: **return** $\mathcal{S}_{abs}$

5.3 Recovering a $\mathbb{G}$-root of an ideal in $\mathcal{L}$ and $\mathcal{T}$ (of Algorithm 5)

An ideal $\hat{\mathtt{I}} \in \mathcal{L}$ has the property that modulo p, i.e. $\mathtt{I} := \hat{\mathtt{I}} + \langle p \rangle$, it is absolutely irreducible.

If the ideal consists of a single point, from Lemma 5.8, each $\mathbb{F}_q$-root of $\mathtt{I}$ lifts to $\mathbb{G}$ where the trivial lifting is the required root.

If the ideal $\mathtt{I}$ has dim > 0 (i.e. the points are 'dense' by Theorem 2.13), then its birationally equivalent hypersurface $\mathtt{H} = \mathbf{V}(h)$ is utilized. Let the map $\psi_2 : \mathbf{V}_{\mathbb{F}_q}(\mathtt{I}) \to \mathtt{H}$ be defined as $(\ell_1, \ldots, \ell_n) \to (\ell_1, \ldots, \ell_r, \ell_0)$, where $r \geq 1$ is the dimension of $\mathbf{V}_{\mathbb{F}_q}(\mathtt{I})$ and ℓ_i's are random linear forms (as in Figure 5.1). Next, consider any lift $\hat{\mathtt{H}} = \mathbf{V}(\hat{h})$ of $\mathtt{H}$ and compute the unique birational equivalence $\hat{\psi}_2 : \mathbf{V}(\hat{\mathtt{I}}) \to \hat{\mathtt{H}}$ and its inverse $\hat{\psi}_1$.

In order to find a root of $\hat{h}$, we pick a random non-singular $\mathbb{F}_q$-root of h, and lift it to a root of $\hat{h}$ (by Lemma 5.7). Finally, use $\hat{\psi}_1 : \hat{\mathtt{H}} \to \mathbf{V}(\hat{\mathtt{I}})$ to get the $\mathbb{G}$-root of $\hat{\mathtt{I}}$ (again by Lemma 5.3), which becomes the required output.

Let $\hat{\mathtt{I}}_0, \ldots, \hat{\mathtt{I}}_{k-2}, \hat{\mathtt{I}}_{k-1} = \hat{\mathtt{I}}$ be the eventual p-adic ideal definitions, in a path of the recursion tree $\mathcal{T}$. An important issue is: We need a $\mathbb{G}$-root common to these ideals. Lemma 5.9 shows that this condition is satisfied by randomly picking a root of $\hat{\mathtt{I}}$. The primality of these ideals togetherwith a 'common' triangular Gröbner basis is key in this proof.

Input: Let $\mathfrak{I} := \{\hat{\mathtt{I}}_0, \ldots, \hat{\mathtt{I}}_{k-1} =: \hat{\mathtt{I}}\}$ be the eventual ideal definitions leading to the leaf $\hat{\mathtt{I}} \in \mathcal{L}$. Ideal $\hat{\mathtt{I}}_\ell \subseteq \widehat{\mathbb{G}}[\mathbf{y}_0, \ldots, \mathbf{y}_\ell]$ is *prime*, for $0 \leq \ell \leq k-1$, and the required prime-power precision is p^k. Further, $\mathtt{I}_\ell := \hat{\mathtt{I}}_\ell + \langle p \rangle$ is absolutely irreducible.

Output: A 'generic' common $\mathbb{G}$-root $(\mathbf{a}_0, \ldots, \mathbf{a}_{k-1})$ of $\mathfrak{I}$, if it exists; ϕ otherwise.

Algorithm 7 Recovering p-adic or $\mathbb{G}$-root common to the ideals $\mathfrak{I}$.

1: **procedure** ROOTS($\mathfrak{I}, p^k$)
2: **if** $\mathbf{V}_{\mathbb{F}_q}(\hat{\mathtt{I}}_{k-1} + \langle p \rangle) = \phi$ **then**
3: **return** ϕ
4: **else if** $\hat{I}_{k-1}$ contains a single point **then**
5: **return** the single point $(\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_\ell)$ (Lemma 5.8).
6: **else**
7: Compute the birationally equivalent hypersurface $\mathtt{H}$ of $\hat{\mathtt{I}}_{k-1} + \langle p \rangle$, over $\mathbb{F}_q$, and the maps $\psi_1 : \mathtt{H} \to \mathbf{V}(\hat{\mathtt{I}}_{k-1} + \langle p \rangle)$ and $\psi_2 : \mathbf{V}(\hat{\mathtt{I}}_{k-1} + \langle p \rangle) \to \mathtt{H}$ using [HW99]. (Also, see Fig. 5.1).
8: Similar to Figure 5.1, compute the hypersurface $\hat{\mathtt{H}}$ birational to $\hat{\mathtt{I}}_{k-1}$ over $\widehat{\mathbb{G}}$, and the mappings $\hat{\psi}_1 : \hat{\mathtt{H}} \to \mathbf{V}(\hat{\mathtt{I}}_{k-1})$ and $\hat{\psi}_2 : \mathbf{V}(\hat{\mathtt{I}}_{k-1}) \to \hat{\mathtt{H}}$.
9: Find a random $\mathbb{F}_q$-root $\mathbf{a}$ on $\mathtt{H}$ using [HW99].
10: Map $\mathbf{a}$ to the 0-th coordinate of the corresponding root of $\hat{\mathtt{H}}$, using Lemma 5.3, to get the approximation $\mathbf{a}'$.
11: Lift $\mathbf{a}'$ to $\hat{\mathbf{a}}$, using Hensel's lifting, which is a root of $\hat{\mathtt{H}}$ modulo p^k (Lemma 5.7).

12: **return** the pullback of $\hat{\mathbf{a}} =: (\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_{k-1})$ given by $\hat{\psi}_1(\hat{\mathbf{a}})$.

5.4 Correctness of HN_{p^k}

Theorem 5.4 (Correctness). *If an ideal in $\mathcal{L}$ returned by Algorithm 5 has $\widehat{\mathbb{G}}$-root $(\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_{k-1})$ (given by Algorithm 7), then the system $\mathcal{F} := \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ has $\mathbb{G}$-root $(\hat{\mathbf{a}}_0 + \ldots + p^{k-1}\hat{\mathbf{a}}_{k-1} \bmod p^k)$. Conversely, if $\mathcal{F}$ has $\mathbb{G}$-root $\mathbf{a}_0 + \ldots + p^{k-1}\mathbf{a}_{k-1}$, then an ideal in $\mathcal{L}$ has some $\widehat{\mathbb{G}}$-root $(\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_{k-1})$, such that $\mathbf{a}_0 + \ldots + p^{k-1}\mathbf{a}_{k-1} \equiv \hat{\mathbf{a}}_0 + \ldots + p^{k-1}\hat{\mathbf{a}}_{k-1} \bmod p^k$.*

Proof. The method of finding $\mathbb{G}$-roots of a system of polynomials goes through three algorithms– Algorithm 5, 6 and 7. The primary Algorithm 5, as discussed, works recursively over a tree; with the branches corresponding to absolutely irreducible components of the $\mathbb{F}_q$-ideals (Step 5 of Algorithm 5 as seen in Section 5.1).

Invariant reduces. We first bound the depth of this tree by showing an easy property: val-multiplicity is at least 1 in every step given by the following lemma.

Lemma 5.5 (Val-multiplicity ≥ 1). *Given the lifting as defined in Algorithm 5, the val-multiplicity is at least 1 in each step; i.e. $p | f_j(\mathbf{y}_\ell + p\mathbf{x}) \bmod \hat{\mathtt{C}}$, where $\hat{\mathtt{C}}$ is the p-adic lift of an irreducible component of the ideal $\mathtt{I}$, as described in the algorithm.*

Thus, at each step of lifting, either a new block $\mathbf{y}_\ell$ is added, or some $\dim(\mathtt{I}_\ell)$ falls. So, the depth of the tree is $\leq k + kn$; and the number of the variables (that store virtual roots) is $\leq nk$. Furthermore, by arguing on the degree and the dimension of the irreducible components, we deduce:

Lemma 5.6 (Size of tree [Dwi22]). *The total number of leaves $\mathcal{L}$ of the recursion-tree $\mathcal{T}$, described in Section 5.1, is at most $d^{(nk)^{O((nk)^2)}}$.*

Roots that lift. We use [HW99, Sec.3] as a subroutine in Steps 3, 7, 14 of Algorithm 6 to obtain the irreducible components of the ideal over $\mathbb{F}_q$. After this, the ideals will be lifted in the main algorithm (Steps 9, 14 of Algorithm 5) using Lemma 5.3. However, its proof needs to create a map from the $\mathbb{F}_q$-ideal (resp. its lift

over $\widehat{\mathbb{G}}$) to a hypersurface. In this process, the map becomes undefined on 'few' roots of H (and hence misses some roots of C in the codomain). This happens because the map $\psi_1 : \mathsf{H} \to \mathbf{V}(\mathsf{C})$ is a *rational* function where the denominators might be zero for some points of H, and (as already discussed) we consider the polynomial e^* which captures these missed 'images' in $\mathbf{V}(\mathsf{C})$. We continue our procedure on $\mathsf{C} + \langle e^* \rangle$ (Step 13 of Algorithm 6). Lemma 5.6 shows that this reduces the dimension of the birationally equivalent hypersurface each time.

Apart from these points, some *singular* points of H might have lifts to $\widehat{\mathbb{G}}$, but we are unable to apply Hensel lifting directly. To cover these roots (and their missed 'images' in $\mathbf{V}(\hat{\mathsf{C}})$), we consider another ideal where we include the pullback, say h^*, of a suitable first-order derivative of the polynomial h (that defines H).

Likewise, when the ideal is *relatively* irreducible, the roots will be shared with h', the derivative of the polynomial representing H, and we add its pullback h^* (Lemma 5.2) in Step 12 of Algorithm 6. The dimension of the ideal thus formed reduces by exactly one [HW99], and we continue decomposing the ideal, until it returns an absolutely irreducible ideal. Thus, the while-loop (Steps 4-14 of Algorithm 6) runs for at most nk steps, which is the maximum possible dimension of the ideal.

Roots captured by $\mathcal{L}$. We want to show that all the roots of $f_1(\mathbf{x}) \equiv \cdots \equiv f_m(\mathbf{x}) \equiv 0 \bmod p^k$ are present in $\mathcal{L}$, and vice versa.

First, we show that roots of $\mathcal{L}$ always give rise to a root of the system $\mathcal{F}$ of polynomial equations $f_1(\mathbf{x}) \equiv \cdots \equiv f_m(\mathbf{x}) \equiv 0 \bmod \langle p^k, \varphi(z) \rangle$. In order to do this, we show the following two lemmas, which help in proving that the roots of the system can be constructed from the roots of the projection of the ideals unto $\mathbb{F}_q$.

Lemma 5.7 (dim > 0 lift)**.** *Given an absolutely irreducible hypersurface H (resp. its lift $\hat{\mathsf{H}}$) over $\mathbb{F}_q$ of* dim > 0*. Its random $\mathbb{F}_q$-root is non-singular with high probability. Thus, we can lift a random root of H to $\widehat{\mathbb{G}}$-root of $\hat{\mathsf{H}}$.*

Lemma 5.8 (single-point lift)**.** *Given an $\mathbb{F}_q$-ideal $\mathtt{I}$ (resp. its lift $\hat{\mathtt{I}}$) that is radical and is a single point. We can uniquely lift it to $\widehat{\mathbb{G}}$-root of $\hat{\mathtt{I}}$.*

As we see, the lifting slightly changes when the ideal consists of a single point.

Now, these single point ideals correspond to 0-dimensional components over $\mathbb{F}_q$, which are $d^{O(nk)}$-many. We can consider the trivial lifts of these single points and check if these finitely many points satisfy the system $\mathcal{F}$.

Using these two lemmas, we prove the correctness of Algorithm 7, which proves one side of the claim: $\mathcal{L}$ exactly captures the roots of the system $\mathcal{F}$.

Lemma 5.9 (Correctness of Algorithm 7)**.** *Given $\widehat{\mathbb{G}}$-ideal $\hat{\mathtt{I}}_{k-1}$ in a leaf of the tree $\mathcal{T}$, Algorithm 7 finds a generic common $\widehat{\mathbb{G}}$-root (if one exists) of the preceding ideals $\{\hat{\mathtt{I}}_\ell \,|\, \ell\}$.*

Using Lemma 5.9, we are now in a position to show that we can recover some root of the system $\mathcal{F}$ from $\mathcal{L}$.

Proposition 5.10 (Root in $\mathcal{L} \longrightarrow$ Root of $\mathcal{F}$)**.** *Given a root of a leaf in $\mathcal{L}$ (using $\mathcal{T}$ and Algorithm 7), we can find a common $\mathbb{G}$-root of the system $\mathcal{F}$ of polynomials f_j, for $j \in [m]$.*

Conversely, we can show that every $\mathbb{G}$-root of the system $\mathcal{F}$ has its p-adic lift present in some ideal of $\mathcal{L}$.

Proposition 5.11 (Root of $\mathcal{F} \longrightarrow$ Root in $\mathcal{L}$, [Dwi22])**.** *If the system of polynomials, as described before, has a root in $\mathbb{G}$, then Algorithm 7 outputs a root for some leaf ideal $\hat{\mathtt{I}}_{k-1}$ in $\mathcal{L}$.*

Therefore, we have shown that the roots of nodes in $\mathcal{L}$ *exactly* correspond to those of $\mathcal{F}$. Further, these can be realized by Algorithm 7. □

Proof of Lemma 5.5. Let us consider the multivariate Taylor's expansion (Definition 2.1) of the j-th polynomial $f_j(\mathbf{y}_\ell + p\mathbf{x})$ given by

$$f_j(\mathbf{y}_\ell + p\mathbf{x}) \;=\; f_j(\mathbf{y}_\ell) + p \sum_{i \in [n]} \partial_{x_i} f_j(\mathbf{y}_\ell) \cdot x_i + \ldots \;, \tag{5.2}$$

where the terms of order-$|\mathbf{i}|$ (partial-derivative) are divisible by $p^{|\mathbf{i}|}$.

As we traverse along the depth of the tree, the polynomial $f_j(\mathbf{y}_\ell) \bmod p$ will be added to $\mathtt{I}$ (Step 4 of Algorithm 5). In Step 5, we consider the absolutely irreducible components of this ideal projected down to $\widehat{\mathbb{G}}/\langle p \rangle = \mathbb{F}_q$, and loop over them from

Step 7 of Algorithm 5. These absolutely irreducible components are such that they are first the factors of $\mathtt{I}$, after which we lift them to $\widehat{\mathbb{G}}$. Thus, for any component $\hat{\mathtt{C}}$ of Step 9, we have $\mathtt{I} = \hat{\mathtt{I}} + \langle p \rangle \subseteq \hat{\mathtt{C}} + \langle p \rangle = \mathtt{C}$.

Now, when we add $f_j(\mathbf{y}_\ell)$ mod p to $\mathtt{I}$, while introducing a new set of virtual roots $\mathbf{y}_\ell$, in Step 4 of Algorithm 5, then Equation 5.2 modulo $\mathtt{I}$ is divisible by p. Therefore, from the previous paragraph, we get that Equation 5.2 modulo $\hat{\mathtt{C}}$ is also divisible by p, implying that the val-multiplicity is ≥ 1, and after division by p, the polynomial will still have coefficients in $\hat{\mathbb{G}}$. □

Proof of Lemma 5.7. Let the hypersurface $\mathtt{H}$ be given by the polynomial $\langle h(Y) \rangle$ over $\mathbb{F}_q(\ell_1, \ldots, \ell_r)$ as before. Since it is absolutely irreducible, the variety $\mathbf{V}(h, h')$ has dimension one less than that of $\mathbf{V}(h)$, where $h' \neq 0$ is some first-order derivative of h. Therefore, the probability of a point being a non-singular root of $\mathtt{H}$, is around $(1 - q^{r-1}/q^r) = 1 - 1/q$ (by Theorem 2.13). Using a random non-singular root, we can lift it to modulo any p-power (by Theorem 3.2); thus, we get a $\widehat{\mathbb{G}}$-root. □

Proof of Lemma 5.8. Since the ideal has a single point say $\mathbf{a}$; the ideal $\hat{\mathtt{I}}$ is just of the form $\langle \mathbf{y} - \hat{\mathbf{a}} \rangle$. So, we output $\hat{\mathbf{a}}$. □

Proof of Lemma 5.9. Using Lemma 5.3, we map an $\mathbb{F}_q$-root of ideal $\mathtt{I}$ to the 0-th coordinate of some (unknown) p-adic root of the ideal $\hat{\mathtt{I}}$. After this, if the root is random, we can lift to a $\widehat{\mathbb{G}}$-root using Hensel's lifting (Lemma 5.7). If the root comes from a single point ideal $\mathtt{I}$, then we use Lemma 5.8. Thus, Algorithm 7 correctly returns a $\widehat{\mathbb{G}}$-root of the given ideal $\hat{\mathtt{I}}$ (as long as, q is 'large' enough for sampling). But what about the other ideals in the path in $\mathcal{T}$ leading to $\hat{\mathtt{I}}$?

We further need to show that the variety $\mathbf{V}(\hat{\mathtt{I}}_{\ell-1})$ extends to $\mathbf{V}(\hat{\mathtt{I}}_\ell)$ (where $\hat{\mathtt{I}}_{\ell-1}, \hat{\mathtt{I}}_\ell$ are the eventual definitions in recursion-tree $\mathcal{T}$). Consider a lift using Lemma 5.3; say in the $\mathbb{F}_q$-ideal $\mathtt{I}_{\ell-1}$ the variables $B_{\ell-1}$ are *localized* giving the triangular form (reduced Gröbner basis) in $\mathbb{F}_q(B_{\ell-1})[B'_{\ell-1}]$, where $B'_{\ell-1} := \cup_{i \leq \ell-1} \mathbf{y}_i \setminus B_{\ell-1}$. Similarly, define B_ℓ and B'_ℓ, for I_ℓ. Recall the variable order, blockwise, $\mathbf{y}_{\ell-1} < \mathbf{y}_\ell$. By Algorithm 5 (Step 8), $B_{\ell-1} = B_\ell$; and $\mathtt{I}_{\ell-1} \subseteq \mathtt{I}_\ell$. Thus, the minpoly of the variables

$B'_{\ell-1}$, over $\mathbb{F}_q(B_\ell)$, in $B_\ell^{-1}\mathtt{I}_\ell$; is the same as it was in $B_{\ell-1}^{-1}\mathtt{I}_{\ell-1}$. So, all the variables $\cup_{i\le\ell-1}\mathbf{y}_i$ have the same minpoly in the two Gröbner bases; and the triangular form is preserved in the new ideal $B_\ell^{-1}\mathtt{I}_\ell$.

Consequently, from Lemma 5.3 lifting, the generators of $B_{\ell-1}^{-1}\hat{\mathtt{I}}_{\ell-1}$ are contained inside those of $B_\ell^{-1}\hat{\mathtt{I}}_\ell$. Therefore, the variety of $\hat{\mathtt{I}}_{\ell-1}$ extends to that of $\hat{\mathtt{I}}_\ell$: For any generic root $(\hat{\mathbf{a}}_0,\ldots,\hat{\mathbf{a}}_\ell)$ of $\hat{\mathtt{I}}_\ell$, the projection $(\hat{\mathbf{a}}_0,\ldots,\hat{\mathbf{a}}_{\ell-1})$ is a root of the predecessor ideal $\hat{\mathtt{I}}_{\ell-1}$.

With this induction step done, we can complete the proof for all $0\le\ell\le k-1$. We use a monomial ordering, that is consistent with all steps, namely: $y_{0,1}<y_{0,2}<\cdots y_{0,n}<\cdots y_{k-1,1}<y_{k-1,2}<\cdots y_{k-1,n}$. Under localization of respective transcendence-basis, we keep the leading term of the generators of these ideals, to be $y_{i,j}$-powers; maintaining the triangular form of Gröbner basis. Thus, a generic root of the leaf $\hat{\mathtt{I}}_{k-1}$, is also a generic common root of the ideals $\{\hat{\mathtt{I}}_0,\ldots,\hat{\mathtt{I}}_{k-1}\}$ that led to the leaf $\hat{\mathtt{I}}_{k-1}\in\mathcal{T}$. □

Proof of Proposition 5.10. We are given a prime ideal, say $\hat{\mathtt{I}}_{k-1}\in\mathcal{L}$, and the associated latest prime ideals $\mathfrak{I}:=\{\hat{\mathtt{I}}_0,\ldots,\hat{\mathtt{I}}_{k-1}\}$ in the recursion-tree $\mathcal{T}$ of Algorithm 5. Let us assume that when the ℓ-th ideal $(\hat{\mathtt{I}}_{\ell-1})$ was defined the last (satisfying Step 8 of Algorithm 5), the j-th polynomial was $f_j^{(\ell)}(\mathbf{x})\in\widehat{\mathbb{G}}[\mathbf{y}_0,\ldots\mathbf{y}_{\ell-1}][\mathbf{x}]$ (done at Step 10 of Algorithm 5).

From the p-adic root of $\mathfrak{I}$, say $(\mathbf{a}_0,\ldots,\mathbf{a}_{k-1})\in\widehat{\mathbb{G}}^{nk}$ by Algorithm 7; we want to show that we can construct a common $\mathbb{G}$-root of $f_j(\mathbf{x})$'s, $j\in[m]$. We prove this by simply using the lifting-steps, one precision at a time, that designed the recursion-tree.

$$\begin{aligned}
&p\mid f_j^{(0)}(\mathbf{y}_0)\bmod\hat{\mathtt{I}}_0\,,\\
&p\mid f_j^{(1)}(\mathbf{y}_0,\mathbf{y}_1)\bmod\hat{\mathtt{I}}_1\,,\\
&\vdots\\
&p\mid f_j^{(k-1)}(\mathbf{y}_0,\mathbf{y}_1,\ldots,\mathbf{y}_{k-1})\bmod\hat{\mathtt{I}}_{k-1}\,.
\end{aligned}\tag{5.3}$$

Next, we can merge the divisibility properties of the key k lifting-steps (Algorithms 5 & 7), at the common p-adic point $(\mathbf{a}_0, \ldots, \mathbf{a}_{k-1})$ of Equation 5.3. This can be written as the following cascading divisibilities:

$$\begin{aligned} p \,|\, f_j^{(0)}(\mathbf{a}_0) \;&\rightarrow p^2 \,|\, f_j^{(0)}(\mathbf{a}_0 + p\mathbf{a}_1) \\ &\rightarrow p^3 \,|\, f_j^{(0)}(\mathbf{a}_0 + p\mathbf{a}_1 + p^2\mathbf{a}_2) \\ &\rightarrow \ \ldots \ \rightarrow \ p^k \,|\, f_j^{(0)}(\mathbf{a}_0 + p\mathbf{a}_1 + \ldots + p^{k-1}\mathbf{a}_{k-1})\,, \end{aligned} \tag{5.4}$$

which provides the required precision p^k; thus giving the $\mathbb{G}$-root $(\mathbf{a}_0 + \ldots + p^{k-1}\mathbf{a}_{k-1})$ of f_j. This finishes proof. □

Chapter 6

Constant degree roots

As we have seen in Section 2.3, factoring a polynomial $f(x) \bmod p^k$ is difficult when $f(x) \equiv \varphi(x)^e \bmod p$, where $\varphi(x)$ is irreducible modulo p, as otherwise, we can find factors using Hensel's lifting. There have been several works trying to factor polynomials modulo prime powers, until [DMS21] gave an algorithm to find the factors for $k \leq 4$. We will call this exponent e as the *transcendence degree*.

The crux of [DMS21] was reducing the problem of factoring a univariate modulo p^k to root finding over the ring $\mathbb{Z}[z]/\langle p^k, \varphi(z)^{ak}\rangle$, where $a \leq e$ is the required transcendence degree of the factor. The reduction has been explained in [Dwi22].

Theorem 6.1 (Reduction [Dwi22])**.** *Let $f(x)$ be a polynomial such that $f(x) \equiv \varphi(x)^e \bmod p$, where $\varphi(x)$ is irreducible over $\mathbb{F}_p$; and $h(x) = \varphi(x)^a - p.y$ be a factor of $f(x) \bmod p^k$ of transcendence degree $a \leq e$, for a polynomial y. Then, $h(x)$ divides $f(x)$ over $\mathbb{Z}/p^k\mathbb{Z}$ if and only if*

$$E = f(x)\cdot(\varphi(x)^{a(k-1)} + \varphi(x)^{a(k-2)}(py) + \ldots + \varphi(x)^a(py)^{k-2} + (py)^{k-1}) \equiv 0 \bmod \langle p^k, \varphi(x)^{ak}\rangle.$$

[Dwi22] further reduces this to root finding of a system of polynomials over $\mathbb{Z}[z]/\langle p^k, \varphi(z)\rangle$. Our algorithm of HN_{pk} from Chapter 5 can thus be used to roots over Galois rings and thereby recover small degree factors of $f(x) \bmod p^k$.

Chapter 7

Conclusions

The problems give some interesting consequences for finding factors of polynomials modulo p^k. However, it is still open to find any factor of $f(x) \bmod p^k$, and progress on this would generalize Hensel's lifting for any factor. A slightly weaker problem would be to find constant degree roots of $f(x) \bmod p^k$ for large k.

Furthermore, the first algorithm of Chapter 3 finds roots of an n-variate d-degree polynomial $f(\mathbf{x}) \bmod p^k$ for small $d, n, \log p$ while that of Chapter 5 does the same for constant n, k. This naturally gives rise to the question of finding roots for constant n, where $d, k, \log p$ can be arbitrarily large.

References

[BLQ13] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin. "Polynomial root finding over local rings and application to error correcting codes". In: *Applicable Algebra in Engineering, Communication and Computing* 24.6 (2013), pp. 413–443.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory*. Vol. 1. Cambridge, MA: MIT Press, 1996.

[Buc65] Bruno Buchberger. "Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal". PhD thesis. Universität Innsbruck, 1965.

[CDS22] Sayak Chakrabarti, Ashish Dwivedi, and Nitin Saxena. "Factoring modular polynomials via Hilbert's Nullstellensatz". submitted. 2022.

[Che+19] Qi Cheng et al. "Counting roots for polynomials modulo prime powers". In: *The Open Book Series* 2.1 (2019), pp. 191–205.

[CLO13] David Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, Switzerland, 2013.

[CS22] Sayak Chakrabarti and Nitin Saxena. "Describing the roots of multivariates mod p^k and efficient computation of Igusa's local zeta function". submitted. 2022.

[CZ81] David G Cantor and Hans Zassenhaus. "A new algorithm for factoring polynomials over finite fields". In: *Mathematics of Computation* 36.154 (1981), pp. 587–592.

[Den84] Jan Denef. "The rationality of the Poincaré series associated to the p-adic points on a variety". In: *Invent. math* 77.1 (1984), pp. 1–23.

[DMS19] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. "Counting Basic-Irreducible Factors Mod p^k in Deterministic Poly-Time and p-Adic Applications". In: *Proceedings of 34th Computational Complexity Conference (CCC 2019)*. Springer, 2019, 15:1–15:29.

[DMS21] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. "Efficiently factoring polynomials modulo p^4". In: *Journal of Symbolic Computation* 104 (2021). Preliminary version appeared in The 44th ACM International Symposium on Symbolic and Algebraic Computation (ISSAC) 2019, pp. 805–823.

[DS20] Ashish Dwivedi and Nitin Saxena. “Computing Igusa’s local zeta function of univariates in deterministic polynomial-time”. In: 14*th Algorithmic Number Theory Symposium (ANTS XIV), Open Book Series* 4.1 (2020), pp. 197–214.

[Dwi17] Ashish Dwivedi. “On the Complexity of Hilbert’s Nullstellensatz over Positive Characteristic”. M.Tech Thesis, Indian Institute of Technilogy, Kanpur, India. 2017.

[Dwi22] Ashish Dwivedi. “Polynomials over composites: Compact root representation via ideals and algorithmic consequences”. PhD thesis. Indian Institute of Technology, Kanpur, India, 2022.

[GCM21] Aditya Gulati, Sayak Chakrabarti, and Rajat Mittal. “On Algorithms to Find p-ordering”. In: *Conference on Algorithms and Discrete Applied Mathematics*. Springer. 2021, pp. 333–345.

[GH96] Joachim van zur Gathen and Silke Hartlieb. *Factorization of polynomials modulo small prime powers*. Tech. rep. University of Paderborn, Germany, 1996, pp. 1–11.

[GH98] Joachim von zur Gathen and Silke Hartlieb. “Factoring Modular Polynomials”. In: *J. Symb. Comput.* 26.5 (1998). Prelimiary version in ISSAC 1996, pp. 583–606.

[Gou97] Fernando Q Gouvêa. *p-adic Numbers*. Springer Berlin, Heidelberg, Germany, 1997.

[GTZ88] Patrizia Gianni, Barry Trager, and Gail Zacharias. “Gröbner bases and primary decomposition of polynomial ideals”. In: *Journal of Symbolic Computation* 6.2-3 (1988), pp. 149–167.

[Hen18] Kurt Hensel. “Eine neue Theorie der algebraischen Zahlen”. In: *Mathematische Zeitschrift* 2.3 (1918), pp. 433–452.

[HW99] M-D Huang and Y-C Wong. “Solvability of systems of polynomial congruences modulo a large prime”. In: *computational complexity* 8.3 (1999), pp. 227–257.

[Igu07] Jun-ichi Igusa. *An introduction to the theory of local zeta functions*. Vol. 14. American Mathematical Soc., Rhode Island, USA, 2007.

[Igu74] Jun-ichi Igusa. “Complex powers and asymptotic expansions. I. Functions of certain types.” In: *Journal für die reine und angewandte Mathematik* (1974).

[Igu77] Jun-Ichi Igusa. “Some observations on higher degree characters”. In: *American Journal of Mathematics* 99.2 (1977), pp. 393–417.

[Kal85] Erich Kaltofen. “Polynomial-time reductions from multivariate to bi-and univariate integral polynomial factorization”. In: *SIAM Journal on Computing* 14.2 (1985), pp. 469–489.

[Kob12] Neal Koblitz. *p-adic Numbers, p-adic Analysis, and Zeta-Functions*. Vol. 58. Springer-Verlag, New York, USA, 2012.

[KT90] Erich Kaltofen and Barry M Trager. “Computing with polynomials given byblack boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators”. In: *Journal of Symbolic Computation* 9.3 (1990), pp. 301–320.

[LN94] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge university press, Cambridge, UK, 1994.

[McD74] Bernard R McDonald. *Finite rings with identity*. Vol. 28. Marcel Dekker Incorporated, New York, USA, 1974.

[Moo93] E Hastings Moore. “A doubly-infinite system of simple groups”. In: *Bulletin of the American Mathematical Society* 3.3 (1893), pp. 73–78.

[NRS17] Vincent Neiger, Johan Rosenkilde, and Éric Schost. “Fast computation of the roots of polynomials over the ring of power series”. In: *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*. 2017, pp. 349–356.

[Pan95] Peter N Panayi. “Computation of Leopoldt’s P-adic regulator.” PhD thesis. University of East Anglia, Norwich, England, 1995.

[Sch74] Wolfgang M Schmidt. “A lower bound for the number of solutions of equations over finite fields”. In: *Journal of Number Theory* 6.6 (1974), pp. 448–480.

[Sir17] Carlo Sircana. “Factorization of polynomials over Z/(pn)”. In: *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*. 2017, pp. 405–412.