DETERMINISTIC ALGORITHMS FOR FACTORS OF CONSTANT-DEPTH ARITHMETIC CIRCUITS

A Thesis
Submitted to the
Tata Institute of Fundamental Research, Mumbai
Subject Board of Technology & Computer Science
for the degree of

DOCTOR OF PHILOSOPHY

by

VARUN RAMANATHAN

School of Technology and Computer Science
Tata Institute of Fundamental Research, Mumbai
Final version submitted in May 2026

DECLARATION

This thesis, titled "Deterministic Algorithms for Factors of Constant-Depth Arithmetic Circuits ", is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgment of collaborative research and discussions.
The work was done under the guidance of Prof. Ramprasad Saptharishi at the Tata Institute of Fundamental Research, Mumbai.

Varun Ramanathan
Mumbai, May 13, 2026

In my capacity as the formal supervisor of record of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

Prof. Ramprasad Saptharishi
Mumbai, May 13, 2026

ACKNOWLEDGMENTS

This is going to be long, get a cup of chai.

Graduate school was super challenging, but the people around me made it incredible, and I would do it all again due to these people.

Ramprasad Saptharishi is the best advisor I could've asked for. He is one of the best teachers I've had, and his enthusiasm (for everything!) is infectious. As an advisor, he was always incredibly patient in so many frustrating meetings where I've stood staring at the board silently. During discussions, he would say "what I'm about to say does not make sense, but let me say it anyway", and thus he taught me to trust my instincts and to not fear mistakes. His questions in seminars are extremely clarifying and leave me wondering why I didn't ask such a question. His constant encouragement and his kindness have been a source of strength. While I do hope to keep collaborating with him, I will sorely miss the ability to knock on his door anytime for a chat.

Mrinal Kumar was not formally my advisor, but he did as much as any advisor would for a student. There is always clarity and honesty in the way he approaches everything. From our walks in the colonnade and at the seaface, his thoughts on how research works (at least for him) have left a lasting impression on me. His work ethic is ever inspiring, and he helped me believe that slowly chipping away at a problem is a viable strategy in research. Indeed, this thesis would arguably not be what it is without his collaboration. I thank him for his support in everything, and hope to keep working with him on more fun projects.

The faculty at TIFR has played a huge role in my understanding of theoretical computer science[1]. Jaikumar taught us one of our first courses at TIFR (Mathematical Foundations of CS), and that course remains a hallmark of what great teaching can be. Anyone who has interacted with Jaikumar would attest to his wisdom and sense of humour, and I've had the good fortune of experiencing it. Arkadev taught us communication complexity as well as query complexity, and his enthusiasm for all things complexity always leaves me wanting to study more of it. I will always remember the couple of hours that he spent with me to help me with some much-needed course correction while I was struggling with communication complexity. Kavitha's wonderful lectures and meticulous notes, Piyush's thought-provoking questions in talks, Prahladh's amazing courses and reading groups (especially in coding theory); these are all parts of the STCS experience that I will carry with me. I unfortunately missed all the chances to take Umang's courses, but I will always dearly remember him for his

1 I take responsibility for the remaining lack of understanding.

kindness and humanity. I thank all the faculty members at STCS for providing such a wonderful environment. I also thank both Arkadev and Akshayaram for being on my thesis committee, for always being deeply curious about the work I was doing, and for their support at all times. The administrative team at STCS consisted of highly competent people who consistently went the extra mile to assist students. I thank Supriya, Sana, Sakshi, Gawade ji, Nitin ji, and Ravi ji for their support. I also thank the administrative staff across TIFR for their assistance. I thank TIFR for providing an environment conducive to research, with supportive people, decent food, beautiful gardens, windy colonnades and our own private seaface. I would also like to thank Prof. Amir Shpilka (Tel Aviv University) and Prof. Chandan Saha (IISc Bengaluru) for their wonderfully detailed and helpful reviews of my thesis.

Research is most fun when it happens with great collaborators, and I felt this first-hand with Shubhangi Saraf, Ben Lee Volk, Shanthanu and Somnath. The last few papers of this thesis happened primarily because Shubhangi expressed her enthusiasm for our previous work and joined us, along with Somnath, for what turned out to be an amazing journey. I thank Shubhangi for the wonderful mentorship. Shanthanu is one of my favourite research-discussion-buddies and I love his attention to detail. His curiosity led us to Furstenberg's paper and the closure results. I hope to be a part of many such happy collaborations.

I spent a couple of incredible months working with Srikanth Srinivasan, Nutan Limaye and Amik Raj Behera at the University of Copenhagen in 2024. Srikanth's impeccable clarity is inspiring and formidable. His explanations cut to the heart of the matter in minutes. Nutan has an eye towards truly understanding the fundamentals, and her exuberance always put me at ease. While I do talk about friends later, this is a good place to thank Amik. None of this would've happened without Amik, the perfect partner-in-crime, who suggested the idea of visiting Srikanth. Our fun table tennis sessions and his Formula One mini-tutorials were an effective antidote to the intensity of our work meetings with Srikanth and Nutan. I hope to continue enjoying such collaborations. I also thank Prateek Dwivedi, Anne Primdahl Kristensen, and all the other members of BARC for the amazing experience. During this visit, I stayed in an Airbnb with the wonderful Anne-Marie, who was extremely kind and joyful. I thank Anne-Marie for her hospitality.

My interest in mathematics and theoretical computer science primarily started because of the environment in IIIT Delhi, and in particular, Rajiv Raman's discrete mathematics course. The joy of problem-solving shone through in all his courses. As my undergraduate project advisor, he reassured me that "speed is not everything in research". Samaresh Chatterjee taught us our first math course – linear algebra. While he is no longer with us, he keeps inspiring us all with his sincerity

and willingness to answer basic questions. Samrith Ram's courses on advanced linear algebra and combinatorics were delightful. Debajyoti Bera, in his awesome courses on theory of computation and complexity theory, would prove theorems without notes, and sometimes, upon getting stuck, think out loud and find his way out of the obstacle. Thus, he inspired me to not be afraid of getting stuck. V. Visweswaran taught us basic electronics in the first semester, and his style of teaching, while unorthodox, emphasised the fundamentals. Sanjit Kaul taught us basic probability and statistics, with an immaculate focus on the basics and a wry sense of humour. I had many such positive experiences in IIIT Delhi, and I thank all the faculty as well as the administrative staff (special mention: Sheetu Ahuja) at IIIT Delhi for the wonderful time. I spent a couple of months of the summer of 2018 in Chennai Mathematical Institute, where I had a lot of fun working on some drone imaging algorithms with Kavita Sutar, and then learning about envy free matchings with Prajakta Nimbhorkar. Some of the friendships from my time in CMI remain to this day [Tanisha, Amit, Ronak, Ramya, Bhargav, et al.]. I spent the summer of 2019 in EPFL, working on FPGA circuit partitioning algorithms with Stefan Nikolic in Paolo Ienne's lab. While I learnt many things during that experience, perhaps the clearest lesson was that I should stick to theory. Paolo was very gracious when I expressed my lack of fit with the lab quite late into the internship, and Ola Svensson generously agreed to discuss some problems in online algorithms for the concluding two weeks at EPFL. It was with Ola that I began to realise the importance of clear articulation during meetings.

I thank all the members of the theory community (both in India and abroad) for being absolutely awesome. I have learnt so much from so many workshops, tutorials, summer schools, winter schools, symposiums, etc. And most importantly, there is an abundance of kindness among theory folks. This community is a gift that never stops giving.

In the last few months, I have had the opportunity to rekindle my mridangam practice under the guidance of Vidwan Anantha R. Krishnan, and it has been absolutely incredible learning with him. The joy of playing the mridangam percolates to all parts of my life, along with other virtues such as patience, consistency and rigour. I thank Anantha for helping me rediscover all of this, and for being so generous with his time, attention, enthusiasm and encouragement. I also thank my original teacher, Kumbakonam Shri N Padmanaban, for being an amazing teacher and for providing a truly encouraging environment for learning the mridangam.

Theory research (and research in general), despite its collaborative nature and friendly community, still requires a scary dose of solitude, with more breakdowns than breakthroughs. I shudder to imagine this journey without my friends and family.

I am extremely fortunate to have met Malavika Sriram. She has always been present and always believed in me, even when I found it impossible to believe in myself. Her company has made everything easier, and it has made life's music sweeter. I hope to enjoy this privilege in the years to come.[2] I also thank Malavika's mother, Vidya Srinivasan, for her love and guidance.

"A set, not a group" is my ever-dependable band of brothers, comprising: Ashutosh, the *prefect compinion* who taught me Tamil and TimeGuessr; Hari, rant-buddy, engineer-at-heart and fellow Dilliwala; and Pranshu, The Walker, who taught me that there is no bridge between Australia and New Zealand. I thank them for being awesome.

While the old guard at STCS was a fantastic bunch [Anamay, Prerona, Neha, Varun, Shubhada, Sayantan, Suhail, et al], I was blessed with an amazing peer group, both at STCS and in other departments. Here's to the crazy ones (with some special names included) – the (old and new) occupants of A-206 [Ashutosh, Pranshu, Hari, Eeshan, Aindrila, Sreejato (Sreejata Kishor Bhattacharya), Agniv], the townsfolk of Ravenswood Bluff (whom I abandoned, but still remain in my heart) [Ratnakar, Bikshan, et al.], the Matunga Fan Club [Malavika, Charvee], the Ragamalika folks (aka Tamil gang), the musicians [Soumyajit, Soumyadeep, Suchith, Binoy, Prajjalak, Shatanik, Jay, Aparajit, et al], the algebra folks [Shanthanu, Soham], the interns [Agastya, Surya, Rohan, Harsh, Parth, Sunaina, Roseline], the quizzers, the colonnade walkers [Binoy], the occasional-chat-in-West-canteen-ers [Koshvendra], running buddies [Sundara] and everyone else! The academic community in India and around the world has given me many beautiful friendships; I thank all my friends for their warmth and for the much-needed dose of familiarity even when I'm halfway around the world [Dhara, Amik, Prashanth, Agastya, Siddhartha, et al.].

Music played a crucial role in the preservation of my sanity. I was lucky to be a part of Harmonic Band and Conduction Band [Prof Vijay, et al.], where I had many unforgettable experiences playing the drums. I also thank the SPIC MACAY Heritage Club for being a source of wonderful classical concerts and art workshops.

I thank all my amazing friends from IIIT Delhi: Anant and Shwetank, my roommates and my metal buddies; the theory gang – Agastya (who has remained a special source of *meowtivation*), Siddhartha, Tharrma, Sagnik, Parth, Ambar, Nihesh, Karamjeet; and everyone else that was a part of my undergrad life.

I did most of my schooling in Venkateshwar International School in Dwarka, Delhi. I have very fond memories of my time in VIS, and I thank my teachers, my friends and the whole school for enriching my childhood. Special thanks: to my English teachers (Madhumita ma'am, Monika ma'am, Anita ma'am, and many others) and the librarians, to whom I owe a lot of my love for stories; to my music teachers

2 Line credit: Anamay :)

(Pawan sir and Vandana ma'am) for providing an environment where every child could learn and enjoy music; to my teachers in eleventh and twelfth grade (Sanjay sir, Giriraj Soni sir and Praveen sir) for their wonderful guidance and teaching. I thank all my school friends [Sidhant, Surbhi, Hemant, Pranjal, Sourav, Sarthak, Abhishek, et al.] who were crucial for my amazing school life. I thank Vansh Sharma, who lived in the same residential complex for most of my childhood and has been a friend since 2006; we have enjoyed many evenings together, playing badminton, cycling around the complex or playing a sophisticated version of hide-and-seek.

I thank my extended family (all the *paatis, thathas, chitthis, chitthappas, perimas, periyappas, atthais, athimbers, mamis, mamas*, cousins, et al.) for their love and support over the years.

I thank Craig Mod, Maria Popova and Oliver Burkeman for their wonderful books and newsletters.

I would never have been at the receiving end of these gifts in life if not for the most important gift, the superpower that gives me an unreasonable advantage: my family. Appa, my father, has taught me courage, that highest virtue which enforces every other virtue. He has taught me the importance of paying attention to one's health. He has gifted me the joy of running and will forever be my favourite running buddy. His unreasonable belief in me never ceases to surprise me. Amma, my mother, is the epitome of unconditional love and support. She is the one who bought me wonderful books at the Scholastic Fair, bought me all kinds of fun toys and mechanic sets (which were sometimes expensive and had to be kept a secret from Appa), and helped me finish boring homework. Amma realised my affinity for percussive instruments and made me join Mridangam classes; Appa would drive me to Mridangam classes after night shifts at the airport. Amma is my movie buddy, with whom I share a thousand Vadivelu and Vivek comedy references. She is the best cook I know, and with her, I have learnt the joy of cooking. And just like Appa, she has taught me to believe in myself by virtue of her own unshakeable belief in me. Finally, my sister Ramya has simultaneously been the greatest source of affection as well as annoyance; this is a conundrum that I am happy to bear for the rest of my life. Her quiet courage in difficult situations has been an inspiration. She is my favourite dancer, and she taught me the joy of dancing. I owe my family everything, and it is to them that I dedicate my thesis.

GENERATIVE AI USAGE : I acknowledge the assistance of ChatGPT in writing TikZ code, in fixing LaTeX errors, and in writing the short verses at the beginning of each of the three parts of the thesis (Ancient History, Recent History, and Present). Everything else, good or bad, is human-generated.

Dedicated to Amma, Appa and Ramya.

LIST OF FIGURES

Figure 1.1 An example of an arithmetic circuit with size 15 and depth 5 2

Figure 1.2 The connections between polynomial factorisation, polynomial identity testing (PIT) and algebraic circuit lower bounds. An arrow from A to B means you can use A to solve B. 8

Figure 1.3 A summary and timeline of our results in polynomial factorisation of constant-depth circuits; the result in green is from [AW24] and included as an important milestone. The final result (in yellow) includes closure results that are not a part of this thesis. 16

CONTENTS

ABSTRACT xix

1 INTRODUCTION 1
- 1.1 Arithmetic circuits 1
- 1.2 Constant-depth arithmetic circuits 2
- 1.3 An overview of multivariate factorisation 3
- 1.4 Closure 5
- 1.5 Deterministic algorithms 6
- 1.6 Factorisation and PIT for constant-depth circuits 8
- 1.7 Our Results 9
- 1.8 The Ideas 13
- 1.9 Organizational remarks 16

I ANCIENT HISTORY: BEFORE ANDREWS–WIGDERSON

2 LOW-DEGREE FACTORS OF CONSTANT-DEPTH CIRCUITS 21
- 2.1 Overview 21
- 2.2 Proof Overview 22
- 2.3 Preliminaries 25
  - 2.3.1 Circuit/formula bit-complexity 26
  - 2.3.2 Relevant subclasses of algebraic circuits 26
  - 2.3.3 Standard preliminaries using interpolation 27
  - 2.3.4 Polynomial identity testing 28
  - 2.3.5 Deterministic divisibility testing and PIT 29
  - 2.3.6 Resultants 30
  - 2.3.7 Hensel Lifting 31
  - 2.3.8 Results on polynomial factorisation 32
- 2.4 Computing candidate low-degree factors of multiplicity one 32
  - 2.4.1 Proof of correctness of the Algorithm 1 34
  - 2.4.2 Running time analysis 35
  - 2.4.3 Proof of structural lemmas 36
- 2.5 Computing factors of all multiplicity 37
- 2.6 Deferred proofs 40

3 LIST-FACTORING 45
- 3.1 Overview 45
- 3.2 Proof Overview 47
  - 3.2.1 Related work and subsequent progress 52
- 3.3 Preliminaries 52
  - 3.3.1 Newton iteration and approximate roots 53
  - 3.3.2 Irreducible polynomials in the field of fractions of a UFD 56
  - 3.3.3 A basic linear algebra fact 57
  - 3.3.4 Irreducibility under a shift of variables 57

3.3.5 Converting a circuit over a number field to a circuit over rational numbers 58
3.4 Pseudo-resultant 59
3.4.1 A starting point for Newton iteration 62
3.5 Generating a list of candidate factors 64
3.5.1 Proof of correctness of Algorithm 3 65
3.5.2 Proof of Theorem 1.7.5 66

II RECENT HISTORY: AFTER ANDREWS–WIGDERSON, BEFORE FURSTENBERG

4 LIST–FACTORING, BUT BETTER 71
4.1 Overview 71
4.2 Preliminaries 71
4.2.1 Squarefree decomposition 72
4.3 High-level sketch of the algorithm 72

5 FACTORS – AND NOTHING ELSE 75
5.1 Overview 75
5.2 Overview of proofs 77
5.2.1 Restricting to roots 78
5.2.2 Analyzing the KI generator on such circuits 78
5.2.3 Computing general factors 80
5.3 Standard preliminaries 81
5.3.1 Truncation 81
5.3.2 Consequences of interpolation 82
5.3.3 Strassen's division elimination 82
5.3.4 Discriminant 83
5.3.5 The Kabanets-Impagliazzo generator 83
5.3.6 Lower bounds for constant-depth circuits 84
5.4 Preliminaries for polynomial factorisation 85
5.4.1 Regularized polynomials 85
5.4.2 Newton iteration 85
5.4.3 A lemma of Chou, Kumar & Solomon 87
5.4.4 Deterministic factorisation 88
5.5 Algorithm 89
5.5.1 Main theorems and high-level overview 89
5.5.2 The algorithm 92
5.5.3 Algorithm for factors of squarefree polynomials 92
5.5.4 Algorithm for obtaining irreducible factors from the right univariate projections 93
5.6 Technical building blocks 94
5.6.1 Properties of roots modulo T^k 94
5.6.2 Complexity of low degree homogeneous components of roots 96
5.7 Warm-up: preserving true roots under variable reduction 100

5.8 Preserving irreducibility under variable reduction 105
5.8.1 From irreducibility testing to divisibility tests 105
5.8.2 From divisibility testing to polynomial identity tests 108
5.8.3 Building up to the proof of Theorem 5.5.1 112
5.8.4 Proof of Theorem 5.5.1 116
5.9 Proof of correctness of the algorithm 117
5.9.1 Analysis of Algorithm 7 117
5.9.2 Analysis of Algorithm 6 122
5.9.3 Analysis of Algorithm 5 123

III PRESENT: POST-FURSTENBERG

6 CLOSURE AND ITS ALGORITHMIC CONSEQUENCES 127
6.1 Overview 127
6.2 Preliminaries 128
6.2.1 Reducing to factorising bivariate polynomials 128
6.2.2 Factorisation of monic polynomials into power series 130
6.2.3 PIT for constant-depth circuits 130
6.3 Closure under taking factors 131
6.3.1 Complexity of power series roots 131
6.3.2 Complexity of general factors 131
6.4 Deterministic algorithms for factorisation 132
6.4.1 Proof of Lemma 6.4.1 135
6.4.2 Deterministic factorisation from hitting-set generators 137
7 FUTURE DIRECTIONS 139

IV ANNEXURES

BIBLIOGRAPHY 143
PUBLICATIONS 153
COLOPHON 155

ABSTRACT

Multivariate polynomial factorisation is a fundamental algebraic problem with connections to error-correcting codes, pseudorandomness and complexity theory. Constant-depth circuits form a restricted yet natural subclass of general polynomial-size arithmetic circuits. They capture a wide variety of algebraic problems. A recent breakthrough of Limaye, Srinivasan and Tavenas gave superpolynomial lower bounds against constant-depth circuits. Combined with a result of Chou, Kumar and Solomon, they obtained the first deterministic subexponential time algorithm for PIT of constant-depth circuits. A result of Kopparty, Saraf and Shpilka reduces the general problem of factoring arithmetic circuits to the problem of PIT, but their connection is not *fine-grained*: the reduction requires PIT for general circuits even for the factorisation of constant-depth circuits. This raises the natural question of obtaining deterministic algorithms for constant-depth circuits using PIT algorithms for constant-depth circuits.

In our first set of results, we obtain a deterministic subexponential-time algorithm for computing all low-degree factors of a constant-depth circuit. We also obtain a quasipolynomial time deterministic algorithm for computing all low-degree factors of a sparse polynomial. A crucial ingredient is a result of Forbes which reduces divisibility testing to PIT. To prove our results, we bound the constant-depth complexity of the PIT instances involved in computing low-degree factors of constant-depth circuits.

In our second set of results, we obtain a deterministic subexponential-time algorithm that outputs a list of circuits containing all the constant-depth factors of a constant-depth circuit, but with two caveats. The output list also contains spurious factors, and the circuits output by the algorithm contain division gates. In some sense, we give a "list-factoring" result, analogous to list-decoding algorithms in coding theory. Building on a beautiful work of Andrews and Wigderson, we obtain a deterministic subexponential-time algorithm that computes all the factors of a constant-depth circuit, with the guarantee that the output list is only polynomially large. This is a straightforward extension of the previous algorithm using Andrews and Wigderson's constant-depth upper bound for the Resultant.

In our third set of results, we obtain a deterministic subexponential-time algorithm that outputs division-free unbounded-depth circuits for exactly all the factors of a constant-depth circuit. Our main structural observation is that the Kabanets-Impagliazzo hitting-set generator, when instantiated with the low-degree hard polynomial of Limaye, Srinivasan and Tavenas, maintains the factorisation pattern of constant-

depth circuits. Surprisingly, the proof of this result works without explicit closure result for constant-depth circuits.

In our final set of results, we obtain a deterministic subexponential-time algorithm that outputs constant-depth circuits for exactly all the factors of a constant-depth circuit. Moreover, the analysis for this algorithm is simpler than that of the previous algorithm, which had weaker guarantees on the output circuits. The correctness of the final algorithm uses the proof of a more fundamental and surprising result: constant-depth circuits are indeed closed under factoring i.e. factors of constant-depth circuits also have constant-depth circuits. Moreover, our analysis proves a fine-grained version of the results of Kopparty, Saraf and Shpilka by reducing factorisation to blackbox PIT for any complexity class that is closed under taking "constant-depth" combinations, such as $\mathsf{VP}, \mathsf{VBP}, \mathsf{VF}$ and VNP.

1 INTRODUCTION

The goal of the thesis is to develop a non-trivially fast deterministic algorithm for the factorisation of constant-depth arithmetic circuits.

In the rest of the chapter, we will elaborate on this goal, build some context for our results and then state our results along with their organization in the thesis.

1.1 ARITHMETIC CIRCUITS

The focus of this thesis will be on algebraic computation. In particular, we will care about multivariate polynomials, and our model for computing multivariate polynomials will be arithmetic circuits. An arithmetic circuit is a non-uniform model of computation that primarily cares about the number of field operations required to syntactically compute a polynomial. We now formally define an arithmetic circuit (also called an algebraic circuit).

Definition 1.1.1 (Arithmetic circuit, Size, Depth)**.** *Let* $\mathbb{F}$ *be a field and let* $n \in \mathbb{N}$ *be the number of variables. An* arithmetic circuit C *is a directed acyclic graph where*

- *each leaf node is labelled by a variable in* $\{x_1, \ldots, x_n\}$ *or an element of* $\mathbb{F}$*,*
- *each non-leaf node is labelled by either an addition* $+$ *or a multiplication* $\times$*,*
- *each edge is labelled by some field constant (to denote multiplication by the field constant), and*
- *computation happens in the natural*[1] *way from leaves to the sinks / output nodes (or the unique sink, which would sometimes be called the* root *node or the* output *node of the circuit).*

The size *of an arithmetic circuit* C*, denoted by* $\mathrm{size}(C)$*, is the number of wires in* C*. The* depth *of an arithmetic circuit* C*, denoted by* $\mathrm{depth}(C)$*, is the length of the longest leaf-to-output path.* ◊

1 To formally define how a circuit – Boolean or arithmetic – performs computation, one can fix an arbitrary topological ordering of the vertices of the directed acyclic graph, and inductively define computation at each vertex. We refer the reader to any standard resource in computational complexity, such as [Golo8].

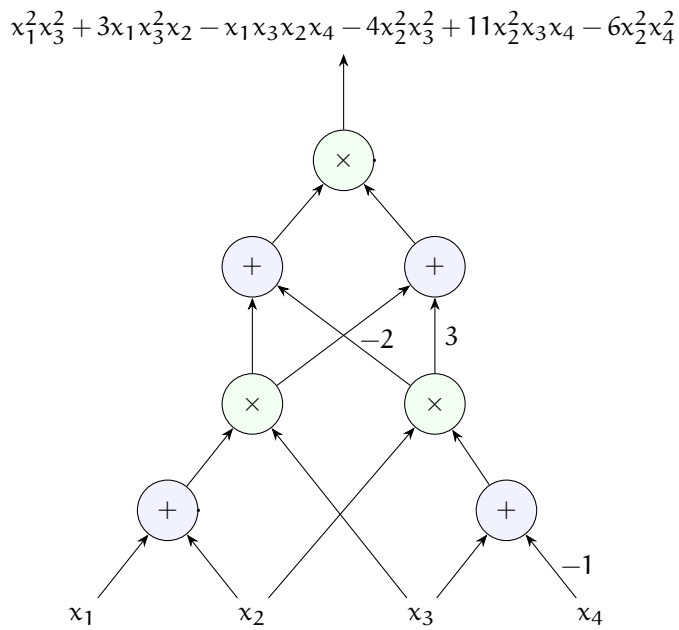

Figure 1.1: An example of an arithmetic circuit with size 15 and depth 5

Any arithmetic circuit computes some multivariate polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ (or a tuple of such polynomials, if there are multiple outputs in the circuit). Moreover, we can represent any polynomial as an arithmetic circuit since the usual *sparse representation* in the monomial basis gives us a depth-2 $\Sigma\Pi$ circuit i.e. a *sum of product of variables*.

For a more detailed exploration of the basics of arithmetic circuits (including the standard definitions of arithmetic complexity classes such as VF, VBP, VP, and VNP[Val79]), we encourage the reader to refer to some of the standard resources such as [SY10; BCS97; Sap14].

1.2 CONSTANT-DEPTH ARITHMETIC CIRCUITS

Consider an infinite family of polynomials $f_1, f_2, \ldots, f_n, \ldots$ such that for each $n \in \mathbb{N}$, f_n is defined on n variables. Such a polynomial family could, for example, be computable by circuits of depth and size given by Table 1.1.

Polynomials:	$f_1(x_1)$	$f_2(x_1, x_2)$	$f_3(x_1, x_2, x_3)$	$\ldots$	$f_n(x_1, \ldots, x_n)$	$\ldots$
Depth:	5	10	15	$\ldots$	$5n$	$\ldots$
Size:	5	20	45	$\ldots$	$5n^2$	$\ldots$

Table 1.1: Polynomial family of depth $5n$ and size $5n^2$.

A different table of circuit size and depth could be given by Table 1.2.

Notice that as the number of variables grows, the depth of the circuit also grows in Table 1.1, whereas the depth remains a fixed

Polynomials:	$f_1(x_1)$	$f_2(x_1, x_2)$	$f_3(x_1, x_2, x_3)$	...	$f_n(x_1, \ldots, x_n)$	...
Depth:	5	5	5	...	5	...
Size:	5	40	135	...	$5n^3$	...

Table 1.2: Polynomial family of depth 5 and size $5n^3$.

constant in Table 1.2. In Table 1.2, we say that the polynomial family is computable by constant-depth circuits. More formally, we have the following definition.

Definition 1.2.1 (Constant-depth arithmetic circuits, VAC_0)**.** *A family of polynomial $\{f_n\}_{n \geqslant 1}$ is said to be computable by constant-depth arithmetic circuits if there exists a natural number $\Delta \in \mathbb{N}$ such that for every $n \in \mathbb{N}$, f_n is computable by an arithmetic circuit of size $\mathrm{poly}(n)$ and depth Δ. We will denote this complexity class by VAC_0, based on the notation in [AW24].* ◊

There are many reasons to care about such a restricted subclass of VP. Once we realise that the usual monomial representation is just a depth-2 circuit, it is natural to consider depth-3, 4 and so on. Many interesting polynomials or polynomial families are known to be computable by polynomial-sized constant-depth circuits. Some classic examples are the elementary symmetric polynomials and power-sum symmetric polynomials. More recently, in a beautiful work, Andrews and Wigderson [AW24] proved that many natural computations on univariate polynomials – such as the GCD, LCM, resultant, etc., – can be computed in VAC_0over fields with characteristic 0. Their work was further generalized in [Bha+25a] to show that these operations can be performed via constant-depth circuits over all sufficiently large fields.

1.3 AN OVERVIEW OF MULTIVARIATE FACTORISATION

While the expressivity of constant-depth circuits makes them interesting to study, there are more reasons to understand the factorisation of constant-depth circuits. But this begs the questions – *why should we care about the complexity of factorisation at all?*

Factorisation is such a natural and fundamental concept that we learn about factoring integers and univariate polynomials in school; this already makes it interesting to study even when divorced from its applications. Unsurprisingly, factorisation has found numerous direct and indirect applications in complexity theory and allied areas. This includes applications to the construction of pseudorandom generators for low degree polynomials [Bog05], algebraic algorithms [KY08], hardness-randomness tradeoffs in algebraic complexity [DSY09; Guo+19], algebraic property testing [PS94; AS03; Ben+20], error correcting codes [Bha+20], deterministic polynomial identity tests for constant-depth circuits [LST25; CKS19] among others.

While the *hardness of integer factoring* is an assumption used in classical cryptography, univariate factorisation over the field of rational numbers is *easy*(!): there is a polynomial time algorithm for factoring univariate rational polynomials, due to Lenstra, Lenstra and Lovasz [LLL82]. Univariate factorisation is quite well-studied and there are other deterministic and randomized algorithms known for univariate factorisation (see [GG13; Sho09] and [BDS25, Section 1.1]). So the next natural question is: what about factors of multivariate polynomials?

Note that for univariate polynomials, the monomial representation (representing the polynomial as a list of monomials with non-zero coefficients, also known as the sparse representation) is already an efficient representation, whereas for multivariate polynomials, the number of monomials can be exponentially large in the number of variables and degree. This is one important reason to consider factorisation of polynomials in the circuit representation: one can still *hope* to achieve algorithms whose running time depends polynomially on the number of variables and degree. Nonetheless, factorisation of multivariate polynomials in the sparse representation is a well-studied problem. We refer the reader to a survey of Roche [Roc18] on algorithms for sparse polynomials.

When multivariate polynomials are represented by arithmetic circuits, a long line of work by von zur Gathen, Kaltofen, Trager, et al. [Gat87; GK85; KT88; Kal85; Kal89; KT90] led to a remarkable breakthrough: there is a randomized polynomial-time algorithm which takes as input an arithmetic circuit and outputs arithmetic circuits for all its factors. This result is remarkable because *a priori*, it is not clear that factors of small arithmetic circuits have small circuits at all, but this is guaranteed by the fact that the algorithm runs in polynomial time. In other words, this result gives us both, a structural result – factors of small circuits have small circuits–, and an algorithmic result – factors of small circuits can be found efficiently. We will refer to structural results of the aforementioned kind as a *closure result*: a class of polynomials $\mathcal{C}$ is *closed under factoring* if whenever $f \in \mathcal{C}$, then every factor $g \mid f$ is also in $\mathcal{C}$.

Once we have Kaltofen's algorithmic result for factoring arithmetic circuits, there are (at least) two natural questions:

1. Can we get similar results for other restricted subclasses of VP? For instance, is VF closed under factoring? If so, can we hope to output formulas for its factors in polynomial time?

2. Since we (complexity theorists) generally believe that randomization cannot buy us too much time, can we get deterministic algorithms for factoring circuits in VP or some subclass of polynomials?

1.4 CLOSURE

For the first question, a simple class to think about is the class of *sparse polynomials* – polynomials where the number of monomials is $\mathrm{poly}(n)$, where n is (always) the number of variables. There is a simple example which shows us that sparse polynomials can have highly non-sparse irreducible factors. Consider the following example due to Forbes and Shpilka [FS15] (inspired by an example of von zur Gathen and Kaltofen [GK85]): the polynomial

$$f(\overline{x}, y) = \prod_{i \in [n]} (x_i^d - 1) + y \cdot \prod_{i \in [n]} (x_i - 1)$$

has $2 \cdot 2^n$ monomials but its irreducible factor

$$g(\overline{x}, y) = \left(y + \prod_{i \in [n]} (1 + x_i + \cdots + x_i^{d-1}) \right)$$

has $d^n + 1$ monomials. Setting, for instance, $d = 2^n$ gives us an example where f has sparsity s but the irreducible factor $g \mid f$ has sparsity $\approx s^{\log s}$. So sparse polynomials are certainly not closed under factoring. While this gives us a lower bound on the sparsity of factors of sparse polynomials, the best upper bound we know is $s^{O(d^2 \log n)}$, where s is the sparsity of the polynomial, n is the number of variables and d is a bound on the individual degree. This upper bound is from a beautiful work of Bhargava, Saraf and Volkovich [BSV18], which also gives a deterministic algorithm that factors n-variate polynomials with sparsity s and individual degree d in time $s^{\mathrm{poly}(d) \log n}$. Since there is a superpolynomial lower bound on the sparsity of factors of sparse polynomials, one could ask, for instance, if factors of sparse polynomials always have small constant-depth circuits; this would be like an *improper learning* statement for factors of sparse polynomials.

For other richer subclasses of VP, there has been partial progress. For algebraic branching programs, Sinhababu and Thierauf [ST20] proved a closure result as well as a randomized polynomial-time algorithm that takes an ABP as its input and outputs ABPs for all its factors. For formulas and constant-depth circuits, Dvir, Shpilka and Yehudayoff [DSY09] proved a root closure result under certain conditions: if $f(\overline{x}, y)$ is an n-variate polynomial with y-degree d, total degree D, and a size s circuit of depth Δ, then we have circuits of size $\mathrm{poly}(D^d \cdot s)$ and depth $\Delta + O(1)$ for any polynomial $g(\overline{x})$ satisfying $f(\overline{x}, g(\overline{x})) \equiv 0$. They proved this result as a step towards hardness vs randomness tradeoffs for constant-depth circuits. Another important piece of work was by Chou, Kumar and Solomon [CKS19], where they built on the ideas of [DSY09] to show that for any n-variate polynomial f with a size-s depth-Δ circuit, every factor of degree d has a circuit of size $\mathrm{poly}(s, d^{\sqrt{d}})$ and depth $\Delta + O(1)$. In other words, low-degree

factors of small constant-depth circuits have small constant-depth circuits. This work eventually played an important role in getting improved hitting-set generators for constant-depth circuits, and we will revisit it soon when we discuss the question of deterministic algorithms for factoring. [CKS19] also proved that VNP is closed under factoring. Other relevant papers include a work of Oliveira [Oli15] – which generalized the work of [DSY09] from roots to factors along with a randomized algorithm to output factors–, and a work of Dutta, Saxena and Sinhababu [DSS22], which proved closure results with quasipolynomial bounds or under some special conditions. All known closure results proceed either via Hensel lifting or via Newton iteration, both of which are iterative procedures for getting better and better approximations for a root/factor. After computing a sufficiently good approximation, these algorithms then reconstruct a true root / factor. The iterative nature of these algorithms is a key obstacle in proving closure results for classes such as VF or VAC_0, where either the size or the depth of the circuit blows up too much in these iterative algorithms. For a comprehensive exploration of closure results for algebraic complexity classes and the underlying techniques, we recommend the survey of Bhargav, Dwivedi and Saxena [BDS25].

1.5 DETERMINISTIC ALGORITHMS

Let us now consider the question of getting deterministic algorithms for factorisation. Progress has been slower on this front, and for good reasons. Shpilka and Volkovich [SV10] gave the following simple reduction from polynomial identity testing to irreducibility testing. Given a polynomial $f(\overline{x})$, let y and z be new variables, and define $g(\overline{x}, y, z) := yz + f(\overline{x})$. Then, $f(\overline{x}) \not\equiv 0$ if and only if $g(\overline{x}, y, z)$ is irreducible. Thus, any deterministic irreducibility testing algorithm (and in particular, any deterministic factorisation algorithm) will also imply a deterministic polynomial identity testing algorithm. In the same work, Shpilka & Volkovich build on these ideas to give an efficient deterministic algorithm for factoring sparse multilinear polynomials. In subsequent works, Volkovich [Vol15; Vol17] gave an efficient deterministic algorithm to factor sparse polynomials that split into multilinear factors and sparse polynomials with individual degree at most 2. More recently, the aforementioned work of Bhargava, Saraf and Volkovich [BSV18] gave a quasipolynomial time deterministic algorithm for factoring sparse polynomials with small individual degree based on some beautiful geometric insights.

In the other direction, a work of Kopparty, Saraf and Shpilka [KSS15] showed a reduction from factorisation to polynomial identity testing: any blackbox (or whitebox) PIT algorithm can be used to solve the blackbox (whitebox) factorisation problem. Thus, an important obser-

vation from [KSS15] is that the use of randomness in factorisation can be narrowed down to the use of randomness for solving PIT.

These works tell us that polynomial factorisation is essentially equivalent to polynomial identity testing. In particular, we shouldn't expect deterministic factorisation algorithms for a class $\mathcal{C}$ that are faster than the best known deterministic PIT algorithm for $\mathcal{C}$. While this greatly clarifies our understanding of polynomial factorisation, it is not particularly great news if we want to derandomize factorisation: derandomization of polynomial identity testing is an extremely well-studied problem with an intimate connection to the (hard) problem of proving explicit algebraic complexity lower bounds. Heintz and Schnorr [HS80] showed, via an elementary linear-algebraic proof, that a hitting set for a class of polynomials can be used to construct polynomials outside that class. In a remarkable result extending the connection from PIT to lower bounds, Kabanets and Impagliazzo [KI04] showed that non-trivial deterministic algorithms for PIT, even in the whitebox setting (where the algorithm has access to a circuit, and not just oracle access), lead to breakthrough lower bounds: either NEXPdoes not have polynomial sized Boolean circuits, or the Permanent does not have polynomial-sized algebraic circuits. In the other direction, building on the seminal work of Nisan and Wigderson [NW97], Kabanets and Impagliazzo [KI04] proved that any explicit hard polynomial for VP can be used to construct a hitting-set generator for VP, where the stretch of the hitting-set generator depends on the hardness of the hard polynomial. For more details, we refer to the survey of Shpilka and Yehudayoff [SY10] or the survey of Kumar and Saptharishi [KS19]. A consequence of this discussion is that obtaining fast deterministic factorisation algorithms for general polynomial sized circuits would imply fast deterministic PIT algorithms for polynomial sized circuits, which in turn would lead to breakthrough lower bounds. Following the previous discussion, one might wonder if deterministic factorisation algorithms are known for restricted circuit classes for which deterministic PIT algorithms are known. Alas, existing results do not live up to this hope. For instance, [KSS15] shows that we can reduce factorisation to PIT, but the PIT instances in their reduction are potentially polynomials in VP even if the input polynomial is, say, a sparse polynomial. Thus, we have polynomial time blackbox interpolation algorithms for sparse polynomials [KS01; BT88], but we do not have even quasipolynomial time algorithms for factoring sparse polynomials (which might be the best we can hope for since we saw that sparse polynomials can have superpolynomially sparse factors). The situation is similar for other restricted classes of circuits. In fact, it is not even clear if one can always get an efficient randomized algorithm (like [KT88]) that takes a circuit from a restricted class $\mathcal{C}$ and outputs circuits in $\mathcal{C}$ for every factor that is in $\mathcal{C}$. We can do this for sparse polynomials: use Kaltofen and Trager's algorithm to factorise the

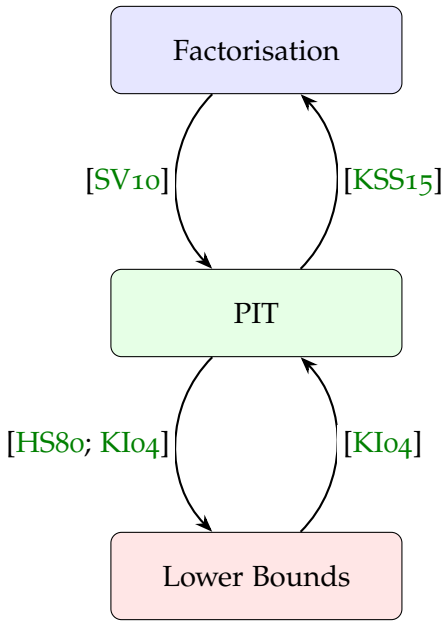

Figure 1.2: The connections between polynomial factorisation, polynomial identity testing (PIT) and algebraic circuit lower bounds. An arrow from A to B means you can use A to solve B.

sparse polynomial in polynomial time, and then use sparse multivariate polynomial interpolation [KS01; BT88] on each factor to compute a potential sparse representation, which can finally be verified easily using randomness.

1.6 FACTORISATION AND PIT FOR CONSTANT-DEPTH CIRCUITS

Let us now talk about VAC_0, the class of polynomials computable by constant-depth circuits. While there has been gradual progress in PIT algorithms for restricted subclasses of VAC_0[DG24], not much attention has been given to the question of deterministically factorising polynomials in these subclasses. A key reason, again, is the lack of closure results for these classes, as well as a lack of understanding of the PIT instances in the standard Hensel lifting / Newton iteration framework for factorisation.

For general constant-depth circuits, the first subexponential time blackbox PIT algorithm came from the breakthrough lower bounds of Limaye, Srinivasan and Tavenas [LST25]. [CKS19] gave closure results in the low-degree regime of parameters (when the degree is about $O(\log n)$), and also showed how to use their result along with appropriate lower bounds (unknown at the time of their work) plugged into the Kabanets-Impagliazzo hitting-set generator to obtain subexponential time PIT for constant-depth circuits. [LST25] gave su-

perpolynomial lower bounds against constant-depth circuits, with the hard polynomial having low degree (about $O(\log n)$). Combining this with the results of [CKS19] gives us the aforementioned PIT algorithm. Subsequently, Andrews and Forbes [AF22] gave a different subexponential PIT algorithm for constant-depth circuits, also based on the lower bounds of [LST25] but better parameters for the complexity of the generator. Again, one might hope that these PIT results would lead to subexponential time algorithms for deterministically factorising constant-depth circuits, but at this point, no such results were known.

It is natural to first think of a *decision* version of the factorisation problem: given polynomials g and f, can we test if $g \mid f$? Clearly the problem is weaker than the factorisation problem. Surprisingly, we did not even know a non-trivial deterministic algorithm to check if a sparse polynomial is divisible by a given constant-degree polynomial, till a work of Forbes [For15]. An important contribution of [For15] is a reduction of divisibility testing to PIT, without the enormous complexity blow-up that happens in [KSS15]. Using this reduction, Forbes converted the question of testing if a constant-degree polynomial divides a sparse polynomial, to a structured depth-4 PIT question, for which he gave a quasipolynomial time deterministic algorithm. More generally, it is easy to observe that if g and f are in VAC_0, then the PIT instance obtained in Forbes's reduction is also in VAC_0.

Forbes's reduction and his PIT results from the same work [For15] essentially form the starting point for our results. While his reduction is from divisibility testing to PIT (and not factorisation to PIT), we will show that the reduction can be extended to factorisation via the standard framework of Hensel Lifting / Newton Iteration.

1.7 OUR RESULTS

We shall now state our results (informally) in chronological order, along with their organization in the thesis.

The first set of results obtain deterministic algorithms that output the constant-degree factors of a constant-depth circuit in subexponential time. For sparse polynomials, the algorithm runs in quasipolynomial time.

Theorem 1.7.1. *[Low-degree factors of constant-depth circuits] Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $d, k \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit C of size s, bit-complexity t, degree D and depth k and outputs all the irreducible factors of C of degree at most d, along with their respective multiplicities in time $(sDt)^{O((sDt)^{\epsilon})}$.

When the input polynomial is sparse, i.e. has a small depth-2 circuit, then the time complexity of the algorithm in Theorem 1.7.1 can be

improved to be quasipolynomially bounded in the input size. This gives us the following theorem.

Theorem 1.7.2. *[Low-degree factors of sparse polynomials] Let* $d \in \mathbb{N}$ *be an arbitrary constant.*

Then, there is a deterministic algorithm that takes as input a polynomial $f \in \mathbb{Q}[\overline{x}]$ *of sparsity* s*, bit-complexity* t*, degree* D*, and outputs all the irreducible factors of* f *of degree at most* d*, along with their respective multiplicities in time* $(sDt)^{\mathrm{poly}(\log sDt)}$.

Remark 1.7.3. *As stated, these results work for extracting constant-degree factors of constant-depth circuits. But, we suspect that with enough care, we can make the proof work for slightly super-constant degree as well, essentially because [LST25] gives us a lower bound of the form* $n^{\omega(D(n))}$ *for a function* $D(n)$ *which grows slightly slower than* $O(\log n)$*. This lower bound is strong enough to give subexponential PIT for constant-depth circuits with slightly super-polynomial size as well, which would show up while instantiating these results for super-constant degree factors. In view of subsequent results, it is perhaps not very valuable to figure out the tightest setting of parameters.* ◊

These results immediately yield an algorithm (with comparable time complexity) to check if the polynomial computed by a given low depth circuit is a product of polynomials of degree at most d. More concretely, we have the following corollary that follows by comparison of degrees of the input polynomial and the low-degree factors (with multiplicities) listed by the algorithms in the above theorems.

Corollary 1.7.4. *Let* $\mathbb{Q}$ *be the field of rational numbers and* $\epsilon > 0$, $d, k \in \mathbb{N}$ *be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit C *of size* s *, bit-complexity* t*, degree* D *and depth* k *and decides if* C *is a product of irreducibles of degree at most* d *in time* $(sDt)^{O((sDt)^{\epsilon})}$.

Moreover, when f *is a sparse polynomial with sparsity* s*, then the algorithm runs in* $(sDt)^{\mathrm{poly}(\log sDt)}$ *time.*

These results are from [KRS23], and we will present these results in Chapter 2.

Our second set of results begins with a subexponential time deterministic algorithm that takes a constant-depth circuit as input and output a subexpontial size list of polynomial size circuits with division gates, such that every constant-depth factor of the input polynomial has a representative in the output list, but there could be spurious factors as well.

Theorem 1.7.5. *[Subexponential list containing constant-depth factors of constant-depth circuits] Let* $\mathbb{Q}$ *be the field of rational numbers and* $\epsilon > 0$, $\Delta_1, \Delta_2 \in \mathbb{N}$ *be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta_1)}$ *of size* s*, bit-complexity* t *and degree* D*, along with a*

size parameter m*, and outputs a list of circuits* $L = \{C_1, \ldots, C_r\}$ *with the following properties.*

1. *Each* $C_i \in L$ *is an arithmetic circuit of size* $\mathrm{poly}(s, m, D)$ *and bit-complexity* $\mathrm{poly}(t, s, D, m)$*, with division gates.*

2. *For every irreducible factor* g *of* C *with a size* m $(\Sigma\Pi)^{(\Delta_2)}$*-circuit computing it, there exists a* $C_i \in L$ *such that* C_i *computes* g*.*

The size of the list L *as well as the running time of the algorithm are bounded above by* $O(smDt)^{O((smDt)^{\epsilon})}$*.*

This result is from [Kum+24] and we present it in Chapter 3.

As a consequence of the beautiful results of Andrews and Wigderson [AW24], we then get the following improved result – a subexponential time deterministic algorithm that takes a constant-depth circuit as input and output a *polynomial* size list of polynomial sized circuits with division gates, such that every factor (and not just every constant-depth factor) of the input polynomial has a representative in the output list, along with potential spurious factors.

A priori, *not every factor of a constant-depth circuit needs to have a constant-depth circuit, but this is true in hindsight, due to the results from Chapter 6.*

Theorem 1.7.6. *[Polynomial-sized list containing constant-depth factors of constant-depth circuits, in subexponential time] Let* $\mathbb{Q}$ *be the field of rational numbers and* $\epsilon > 0$*,* $\Delta \in \mathbb{N}$ *be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta)}$ *of size* s*, bit-complexity* t *and degree* D*, and outputs a list of circuits* $L = \{C_1, \ldots, C_r\}$ *with the following properties.*

1. *Each* $C_i \in L$ *is an arithmetic circuit of size* $\mathrm{poly}(s, D)$ *and bit-complexity* $\mathrm{poly}(t, s, D)$*, with division gates.*

2. *For every irreducible factor* g *of* C*, there exists a* $C_i \in L$ *such that* C_i *computes* g*.*

The size of the list L *is* $\mathrm{poly}(d)$*. The running time of the algorithm is bounded above by* $O(sDt)^{O((sDt)^{\epsilon})}$*.*

We present a sketch of this improved list-factoring theorem in Chapter 4.

The next result obtains the cleanest improvement one could imagine without getting a closure result: a deterministic subexponential time algorithm that takes a constant-depth circuit as input and outputs polynomial-sized (division-free) circuits for exactly all the factors of the input polynomial.

Theorem 1.7.7. *[Deterministic subexponential time algorithm for factorising constant-depth circuits] For every constant* $\epsilon > 0$*, constant* $\Delta \in \mathbb{N}$ *and sufficiently large* n*, the following statement is true.*

There is a deterministic algorithm that takes as input an algebraic circuit C *over the field* $\mathbb{Q}$ *on* n *variables with depth* Δ*, and size and degree* $\mathrm{poly}(n)$*, runs in time* $\exp(O(n^{\epsilon}))$ *and outputs algebraic circuits (of potentially unbounded depth) for all irreducible factors of* C*, together with their multiplicities.*

This result is from [Bha+25c] and we present it in Chapter 5.

Before we state our final set of results, we first state a closure theorem proven in [Bha+25b]. The theorem states that constant-depth circuits, formulas and essentially any circuit class that is closed under taking "constant-depth" combinations, are closed under factoring.

Theorem 1.7.8. *[Closure under factorisation [Bha+25b]] Let $\mathbb{F}$ be a field of characteristic zero, f be a polynomial on n variables of degree d over $\mathbb{F}$ and g be a factor of f. Then, the following are true.*

If f can be computed by an algebraic circuit of size s and depth Δ over $\mathbb{F}$, then g can be computed by an algebraic circuit of size $\mathrm{poly}(s, d, n)$ and depth $\Delta + O(1)$ over $\mathbb{F}$.

If f can be computed by an algebraic formula of size s over $\mathbb{F}$, then g can be computed by an algebraic formula of size $\mathrm{poly}(s, d, n)$ over $\mathbb{F}$.

As a consequence of the proof of the closure result, we obtain stronger guarantees for the aforementioned deterministic algorithm – the algorithm outputs formulas or circuits (depending on the input) with an $O(1)$ increase in depth–, along with a much simpler analysis.

Theorem 1.7.9. *Let $\mathbb{F}$ be the field of rational numbers. Fix any constant $\Delta \in \mathbb{N}$ and $\varepsilon > 0$. Then, there is a deterministic algorithm $\mathcal{A}_{\Delta,\varepsilon}$ that takes as input a size s depth-Δ circuit for a degree d polynomial $P(\overline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ and outputs circuits of size $\mathrm{poly}(s, d)$ and depth $\Delta + O(1)$ for each irreducible factor $g(\overline{x})$ of $P(\overline{x})$, along with their multiplicities. Moreover, $\mathcal{A}_{\Delta,\varepsilon}$ runs in time $\mathrm{poly}(s, d)^{O(n^{\varepsilon})}$.*

This result is from [Bha+25b] and we present it in Chapter 6. While we won't include a proof of Theorem 1.7.8 as a part of this thesis, we will include a proof of Theorem 1.7.9 using the closure results.

BIT-COMPLEXITY IN CHAPTERS 4 TO 6: While Theorems 1.7.6, 1.7.7 and 1.7.9 do not explicitly mention how the bit-complexity of the input circuit shows up in the running time or the complexity of the factors, the operations in our algorithms/results are essentially all the same – e.g. interpolation, evaluation of formulas/constant-depth circuits, etc., – and with the same proofs, we can show that the bit-complexities of all the rational numbers and the circuits that show up along the way remain polynomially bounded in the input parameters.

DEPENDENCE OF RUNNING TIME ON DEPTH: The running time of our algorithms *do not* significantly depend on the depth of the input circuit. The running time of these factorisation algorithms is essentially the same as the running time of polynomial identity testing for relevant circuit classes. The proof from [KI04], when instantiated with the superpolynomial lower bounds from [LST25] along with the required closure results [CKS19; Bha+25b], leads to a running time of $s^{O(n^{\varepsilon})}$ for *any* $\varepsilon > 0$, regardless of the depth of the input circuits. Our

results, like most of complexity theory, are asymptotic in nature, and the input length for which an algorithm's guarantees kick in might subtly depend on the depth of the circuit being considered.

1.8 THE IDEAS

We shall now give an overview of the main ideas in our proofs. Consider the following outline which is used in all known circuit factorisation algorithms.

1. **Preprocessing**: Ensure that the polynomial $f(\overline{x}, y)$ is monic, squarefree, etc.
2. **Projection**: Map $f(\overline{x}, y)$ to a univariate polynomial $F(y)$ while preserving some information about factors. For instance, set $\overline{x} = \overline{\alpha}$ for some $\overline{\alpha} \in \mathbb{F}^n$ which ensures that the (squarefree) polynomial remains squarefree after the projection.
3. **Univariate factorisation**: Factorise the univariate projection $F(y)$ using an algorithm appropriate for the setting (underlying field, randomized vs deterministic, etc.)
4. **Lifting**: Use Newton iteration or Hensel lifting to lift the univariate roots or factors to *approximate* roots or factors of the polynomial.
5. **Recovery**: From a sufficiently good *approximate* root/factor, recover an original factor. Typically done by solving an appropriate linear system.

Conceptually, an important bottleneck for derandomization is the second step. At the heart of all our results, we have some structural observations which give us a way of deterministically finding a projection that preserves some information about factors.

LOW-DEGREE FACTORS OF CONSTANT-DEPTH CIRCUITS: In our first set of results, we obtain a deterministic algorithm for computing the low-degree (constant-degree) irreducible factors of a constant-depth circuit. Intuitively, we would like to ensure that low-degree factors do not "mix" with the other factors after we project to the univariate setting. More formally, if $f(\overline{x}, y) = g(\overline{x}, y) \cdot h(\overline{x}, y)$, where g is a low-degree irreducible polynomial and $\gcd(g, h) = 1$, we would like to find $\overline{\alpha} \in \mathbb{F}^n$ so that $\gcd(g(\overline{\alpha}, y), h(\overline{\alpha}, y)) = 1$. The Resultant of g and h captures this gcd condition: the Resultant of $g(\overline{x}, y)$ and $h(\overline{x}, y)$ (with respect to the variable y) is a non-constant polynomial if and only if their gcd is non-trivial. By finding a point $\overline{\alpha}$ which ensures that the non-constant Resultant remains non-constant, we can ensure that $\gcd(g(\overline{\alpha}, y), h(\overline{\alpha}, y)) = 1$. Crucially, when g is a low-degree

factor and f has a constant-depth circuit, this Resultant provably has a constant-depth circuit. Thus, we can use hitting-set generators for constant-depth circuits to find a suitable projection.

AN ATTEMPT TO FIND ALL CONSTANT-DEPTH FACTORS: Suppose g is an irreducible factor of f, where both f and g have constant-depth circuits and $g \nmid (f/g)$. At this point, it is not too hard to show that the quotient $h = f/g$ also has a constant-depth circuit. Taking a hint from the aforementioned outline of factorisation algorithms, one could try and ask for a sufficient condition to ensure that Newton iteration can be used for recovering g. In Newton iteration, we start with a (scalar) root of the univariate projection of g, lift it to an *approximate* (polynomial) root of g, and then recover g from it. Intuitively, if we could find some $\overline{\alpha}$ such that $g(\overline{\alpha}, y) \nmid h(\overline{\alpha}, y)$ – i.e. the univariate projection of g does not get subsumed by the univariate projection of h – then we can say that we have preserved some information about g. More formally, we can prove that Newton iteration works if the univariate projection of g has a root of multiplicity one[2], which is not a root of the univariate projection of h. One way to satisfy this condition is via the Resultant of g and h: by ensuring that the Resultant of g and h remains non-trivial post-projection, we can ensure that $\gcd(g(\overline{\alpha}, y), h(\overline{\alpha}, y))$ is trivial, and in particular, $g(\overline{\alpha}, y) \nmid h(\overline{\alpha}, y)$. Unfortunately, at the time of [Kum+24], it wasn't known that the Resultant of two constant-depth circuits has a constant-depth circuit (which follows from [AW24]). Thus, this reduction to PIT was not known to be a reduction to PIT for constant-depth circuits. But notice that the Resultant tries to maintain the triviality of $\gcd(g(\overline{x}, y), h(\overline{x}, y))$, which is stronger than our requirement of finding some $\overline{\alpha}$ such that $g(\overline{\alpha}, y) \nmid h(\overline{\alpha}, y)$. Just by itself, our requirement is a divisibility preservation condition, and using Forbes's idea, we can convert this to a PIT instance (which is in VAC_0 when g and f are in VAC_0) and find a good projection $\overline{\alpha}$.

Since we don't actually know the factors, we can go over all the projection options from the hitting set (for each correct factor $g \mid f$, there exists a good projection $\overline{\alpha}_g$), and then try to do a divisibility test in the end to weed out spurious factors. The problem with the divisibility test is that Newton iteration leads to a blow-up in depth and the candidate factors need not have constant-depth circuits. In particular, using a hitting-set generator for constant-depth circuits need not work to check the divisibility of candidate factors. Thus, the best we can do is output a list of unbounded-depth circuits with some circuit for each irreducible factor, but also some circuits for spurious factors.

Since the resulting algorithm runs in subexponential time, the output list could be subexponentially large. When we use the results in

2 The multiplicity one requirement is similar to how Newton iteration does not work at stationary points.

[AW24], we can guarantee that the list will be at most polynomially large; the reason for this is explained in Chapter 4.

EXACTLY ALL THE FACTORS: Recall that in the factorisation algorithm, we first compute approximate roots for the (pre-processed) input polynomial (these are truncations of power series roots), and then we recover the actual factor by solving a linear system to find the *minimal polynomial* for the approximate root. If $\phi_1(\overline{x}), \ldots, \phi_d(\overline{x})$ are the power series roots, then each factor is equal to $\prod_{i \in S}(y - \phi_i(\overline{x}))$ for some subset $S \subseteq [d]$. Moreover, if $\widetilde{\phi_1}(\overline{x}), \ldots, \widetilde{\phi_d}(\overline{x})$ are approximate roots of sufficient accuracy, then it is easy to show that each factor is equal to $\left[\prod_{i \in S}(y - \widetilde{\phi_i}(\overline{x}))\right]_{\leqslant D}$ for some subset $S \subseteq [d]$ and some truncation parameter D. In other words, if we knew the right partition of roots into conjugates, then it is easy to recover factors. In [Bha+25c], we show that a specific hitting-set generator for constant-depth circuits, when used as a variable-reduction map, ensures that conjugate roots stay together, i.e. it preserves the irreducibility of each factor. The specific hitting-set generator is obtained by plugging in the hard polynomial from [LST25] into the hardness vs randomness framework of Kabenets and Impagliazzo [KI04], aided by the structural results of [CKS19]. To prove this, we show that irreducibility testing reduces to PIT. Curiously, the PIT instances that show up in this reduction were not known to be constant-depth circuits at the time of [Bha+25c]; nonetheless, we were able to show that a hitting-set generator for constant-depth circuits preserves the nonzeroness of these instances.

The algorithm for obtaining circuits for exactly all irreducible factors proceeds as follows. After the preprocessing step, we use the [KI04; LST25; CKS19] hitting-set generator to project the input polynomial to a low-variate setting and then use a dense-representation factorisation algorithm to figure out the right partition of approximate roots into conjugates. Then, we can compute these approximate roots via Newton iteration and combine them (using simple products and truncation) as mentioned earlier.

SIMPLER PROOF AND STRONGER GUARANTEES: The results of [Bha+25b] show that factors of constant-depth circuits have constant-depth circuits. The proof proceeds by first showing that approximate roots of a polynomial can be computed with only a constant-depth overhead, using a result of Furstenberg [Fur67], which gives an explicit and succinct expression for power-series roots of polynomials. Thus, the PIT instances that show up in the reduction from irreducibility testing to PIT, also end up being constant-depth circuits. With this proof, we show that *any* hitting-set generator for constant-depth circuits (and not just the one using [KI04; LST25; CKS19]) will preserve the irreducibility of factors of constant-depth circuits. Moreover, the algorithm mentioned in the previous paragraph can now output

constant-depth circuits for each irreducible factor since each factor is formed by a simple combination of approximate roots (which now have constant-depth circuits that can be efficiently computed).

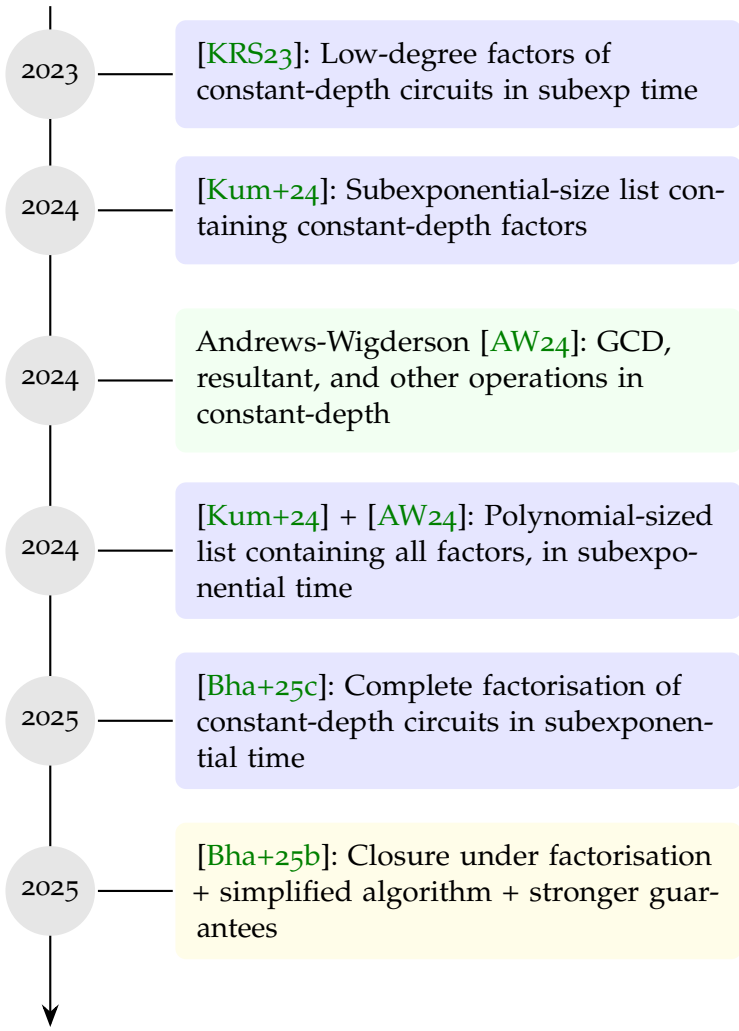

Figure 1.3: A summary and timeline of our results in polynomial factorisation of constant-depth circuits; the result in green is from [AW24] and included as an important milestone. The final result (in yellow) includes closure results that are not a part of this thesis.

1.9 ORGANIZATIONAL REMARKS

- There is a lot of overlap in the preliminaries required for each chapter. We introduce most of these preliminaries in Chapter 2, and then we introduce new preliminaries in each chapter whenever we require it. Thus, we avoid too many repetitions without having a separate chapter covering all the preliminaries.

- Something slightly peculiar about this thesis is that the results of each chapter are subsumed by the results of subsequent chapters. Nonetheless, we present all the results in their chronological order to present a story of how progress happened. As a consequence, some parts of a chapter might refer to, say, certain open problems, which might be solved in a subsequent chapter. We retain these temporal artifacts with explanatory comments where necessary.

Part I

ANCIENT HISTORY: BEFORE ANDREWS-WIGDERSON

In shadow lay the secret of roots,
Where factors hid in circuits deep.
Many sought, but few could find
The path through darkness to what binds.

2 LOW-DEGREE FACTORS OF CONSTANT-DEPTH CIRCUITS

2.1 OVERVIEW

As mentioned in the introduction, till a work of Forbes [For15], we did not even know a non-trivial deterministic algorithm for testing whether a given constant-degree polynomial divides a given sparse polynomial. The first step in our journey will be the following theorem (as stated in the first chapter), which gives a deterministic subexponential time algorithm that takes a constant-depth circuit as input and outputs all constant-degree factors (where the constants are prespecified to the algorithm).

Theorem 1.7.1. *[Low-degree factors of constant-depth circuits] Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $d, k \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit C of size s, bit-complexity t, degree D and depth k and outputs all the irreducible factors of C of degree at most d, along with their respective multiplicities in time $(sDt)^{O((sDt)^{\epsilon})}$.

We note that the bit-complexity of an algebraic circuit/formula is a measure of the bit-complexities of the rational numbers appearing in the circuit. See Definition 2.3.1 for a formal definition.

When the input polynomial is sparse, i.e. has a small depth-2 circuit, then the time complexity of the algorithm in Theorem 1.7.1 can be improved to be quasipolynomially bounded in the input size. This gives us the following theorem.

Theorem 1.7.2. *[Low-degree factors of sparse polynomials] Let $d \in \mathbb{N}$ be an arbitrary constant.*

Then, there is a deterministic algorithm that takes as input a polynomial $f \in \mathbb{Q}[\overline{x}]$ of sparsity s, bit-complexity t, degree D, and outputs all the irreducible factors of f of degree at most d, along with their respective multiplicities in time $(sDt)^{\mathrm{poly}(\log sDt)}$.

These results immediately yield an algorithm (with comparable time complexity) to check if the polynomial computed by a given low depth circuit is a product of polynomials of degree at most d. More concretely, we have the following corollary that follows by comparison of degrees of the input polynomial and the low-degree factors (with multiplicities) listed by the algorithms in the above theorems.

Corollary 2.1.1. *Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $d, k \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit C *of size* s *, bit-complexity* t*, degree* D *and depth* k *and decides if* C *is a product of irreducibles of degree at most* d *in time* $(sDt)^{O((sDt)^{\epsilon})}$.

Moreover, when f *is a sparse polynomial with sparsity* s*, then the algorithm runs in* $(sDt)^{\mathrm{poly}(\log sDt)}$ *time.*

Field dependence of our results

We end this section with a remark about the field dependence of our results. The field dependence in our results stems from two reasons. We need an efficient deterministic algorithm for factorisation of univariate polynomials over the underlying field $\mathbb{F}$. In addition to this, our proofs also need non-trivial deterministic algorithms for polynomial identity testing (PIT) for constant-depth circuits (or very special depth-4 circuits for Theorem 1.7.2) over the underlying field.

The field of rational numbers satisfies both these requirements: a classical algorithm of Lenstra, Lenstra and Lovász[LLL82] solves the problem of deterministic univariate factorisation efficiently over $\mathbb{Q}$ and a recent work of Limaye, Srinivasan and Tavenas [LST25] gives a subexponential time deterministic algorithm for PIT for constant-depth circuits over $\mathbb{Q}$. For Theorem 1.7.2, the relevant PIT is for special depth-4 circuits and was given in a work of Forbes [For15]. In fact, Forbes' result holds even over finite fields.

We restrict our attention to just the field of rational numbers in the presentation although our results work over any large characteristic field that supports the above requirements.

Related work

In a recent paper [DbT24], Dutta, Sinhababu and Thierauf generalized and simplified our results. Along with some other results, they showed that a whitebox PIT algorithm is sufficient to compute low-degree factors of a constant-depth circuit, whereas our reduction requires blackbox PIT algorithms.

2.2 PROOF OVERVIEW

We now give an overview of some of the main ideas in our proofs. In a nutshell, our proofs are based on relatively simple structural observations on top of the existing factoring algorithms. The key is to understand the structure of circuits for which we need a PIT algorithm at every step a little better, and when looking for low-degree factors, we observe that these PIT instances are relatively simple and their circuit complexity is comparable to the circuit complexity of the input polynomials themselves. We also crucially use the divisibility testing

idea of Forbes [For15] in our algorithm at two stages; this helps us handle factors of large multiplicities and also lets us obtain true factors from the output of Hensel Lifting step of the factorisation algorithms. This idea again helps in reducing the complexity of the PIT instance we face in these steps, and in particular, we completely avoid the linear systems solving step in a typical factorisation algorithm that naively (e.g. see [KSS15]) seems to require PIT for algebraic branching programs. Once the PIT instances are shown to be relatively simple, we invoke the PIT algorithms of Forbes [For15] and Limaye, Srinivasan & Tavenas [LST25] to solve these deterministically.

TYPICAL STEPS IN A POLYNOMIAL FACTORISATION ALGORITHM: Most factorisation algorithms (and ours, modulo minor deviations) follow this template:

1. **Making f monic:** Apply a suitable transformation of the form $x_i \mapsto x_i + \alpha_i y$ to ensure that f is monic in y. We may now assume that $f \in \mathbb{Q}[\overline{x}, y]$.
2. **Preparing for Hensel lift:** Ensure that $f(\overline{x}, y)$ is square-free, and further that $f(\overline{0}, y)$ is also square-free.
3. **Univariate factorisation:** Factorise the univariate polynomial $f(\overline{0}, y)$ as a product $g_0(y) \cdot h_0(y)$ where $\gcd(g_0, h_0) = 1$. This can be intepreted as a factorisation $f(\overline{x}, y) = g_0 \cdot h_0 \bmod \mathcal{I}$ where $\mathcal{I} = \langle \overline{x} \rangle$.
4. **Hensel lifting:** Compute an iterated lift to obtain $f = g_\ell \cdot h_\ell \bmod \mathcal{I}^{2^\ell}$ for a suitably large ℓ.
5. **Reconstruction:** From g_ℓ, obtain an honest-to-god factor g of f (unless f is irreducible).

The first two steps typically involve the use of randomness for suitable polynomial identity tests. In the first step, we would like $\overline{\alpha}$ to be a point that keeps the highest degree homogeneous component of f non-zero, and the second step is handled by translating f by a point $\overline{\delta}$ that keeps the "discriminant" of f non-zero. The Hensel lift is a deterministic subroutine that eventually yields small circuits for the lifted factors and the reconstruction step typically involves solving a linear system. It is mostly due to the "discriminant" that we do not have efficient deterministic factorisation algorithm even for constant-depth circuits as the best upper bound for the discriminant we have is an algebraic branching program and we do not have efficient hitting sets for them.

Due to [AW24; Bha+25a], we now have constant-depth circuits for the discriminant, and this was important for the results in Chapter 4, Chapter 5, and Chapter 6.

For our case, it is instructive to focus on a specific factor g of f and understand what would be required to make the above template yield this factor. The first observation is that the base case of Hensel Lifting does not require f to be square-free but rather that the factor g we intend to reconstruct satisfies $g|f$ and $g^2 \nmid f$. For now, let us assume

this and also that f (and hence g and h also) is monic in y. We have that $\gcd(g, h) = 1$ but for the Hensel lift, we also need to find a $\overline{\delta}$ that ensures that $\gcd(g_0, h_0) = 1$ where $g_0 = g(\overline{\delta}, y)$ and $h_0 = h(\overline{\delta}, y)$. The set of "good" $\overline{\delta}$'s is precisely the points that do not make the resultant $\mathrm{Res}_y(g, h)$ zero and thus we want to understand the circuit complexity of this resultant.

The resultant $\mathrm{Res}_y(g, h)$ is the determinant of a matrix of dimension $O(\deg_y(g) + \deg_y(h))$ and its entries are coefficients of g, h when viewed as univariates in y. However, we are only given that f is computable by a constant-depth formula and we do not have any good bound on the complexity of h. We circumvent this by working with a *pseudo-quotient* (introduced by Forbes [For15] in the context of divisibility testing) $\tilde{h}$ of f and g; we work with $\mathrm{Res}_y(g, \tilde{h})$ and show that it is also computable by constant-depth circuits of not-too-large size. Fortunately, the result of Limaye, Srinivasan and Tavenas [LST25] yields sub-exponential sized hitting sets for constant depth formulas and that enables us to avoid the use of randomness to prepare for the Hensel Lifting step.

We can then factorise the univariate polynomial $f(\overline{\delta}, y)$ and attempt all possible factors g_0 of degree at most d to begin the lifting process from $g_0 \cdot h_0$ (where $h_0 = f(\overline{\delta}, y)/g_0$). After an appropriately large lift, we have small circuits (of possibly unbounded depth) computing g_ℓ and h_ℓ such that $\tilde{f} = f(\overline{x} + \overline{\delta}, y) = g_\ell \cdot h_\ell \bmod \mathcal{I}^{2^\ell}$. If g_ℓ is guaranteed to be monic, and the initial choice of g_0 was indeed $g(\overline{\delta}, y)$, the uniqueness of Hensel lifting would ensure that g_ℓ is indeed equal to g (after truncating higher order terms). We can then use standard interpolation to obtain g_ℓ explicitly written as a sum of monomials. Finally, to ensure that g_ℓ is indeed a legitimate factor of $\tilde{f}$, we perform divisibility testing to check if $g_\ell \mid \tilde{f}$.

HANDLING FACTORS OF LARGE MULTIPLICITY: The above overview is all we need to obtain any factor g of degree $O(1)$ that divides f with $g^2 \nmid f$. In order to handle factors with higher "factor-multiplicity", we use a simple observation that $g^{a-1} \mid f$ but $g^a \nmid f$ if and only if g divides $f, \partial_y f, \ldots, \partial_{y^{a-1}} f$ but not $\partial_{y^a} f$. We run our algorithm for each of the partial derivatives to collect the list of candidate factors, and eventually prune them via appropriate divisibility tests.

THE SPECIFIC CASE OF $\Sigma\Pi$-FORMULAS (OR SPARSE POLYNOMIALS): The above sketch yields a sub-exponential time algorithm for obtaining $O(1)$-degree factors of constant depth formulas. However, with some additional care, we obtain a quasipolynomial time algorithm in the case when f is a sparse polynomial. The key observation for this is that we do not really need f to be made monic for the above approach, but we only need g to be monic to exploit the uniqueness of Hensel lifts. Since g is a polynomial of degree at most $d = O(1)$, we can find

a *low Hamming weight* vector $\overline{\alpha}$ such that $g(\overline{x} + y\overline{\alpha})$ is monic in y. This allows us to control the sparsity increase of f in the process and we show that the relevant resultant is a polynomial of the form

$$\sum_i \text{monomial}_i \cdot (O(1)\text{-degree})^{e_i}.$$

Forbes [For15] shows that there are quasipolynomial size hitting sets for such expressions and we use this instead of the more general hitting set of Limaye, Srinivasan and Tavenas [LST25].

2.3 PRELIMINARIES

This section consists of all the necessary building blocks to describe and analyse (in Section 2.4) the main algorithm.

Notation

1. Throughout this chapter (and the rest of the thesis), we work over the field $\mathbb{Q}$ of rational numbers. For some of the statements that are used more generally, we use $\mathbb{F}$ to denote an underlying field.
2. We use lower case letters with an overline like $\overline{x}, \overline{y}, \overline{a}$ to denote tuples, e.g. $\overline{x} = (x_1, x_2, \ldots, x_n)$. The arity of the tuple is either stated or will be clear from the context.
3. For a polynomial f and a non-negative integer k, $\text{Hom}_k[f]$ denotes the homogeneous component of f of degree *equal* to k. $\text{Hom}_{\leqslant k}[f]$ denotes the sum of homogeneous components of f of degree at most k, i.e.,

$$\text{Hom}_{\leqslant k}[f] := \sum_{i=0}^{k} \text{Hom}_i[f].$$

4. The *sparsity* of a polynomial f is the number of monomials with a non-zero coefficient in f.
5. We will refer to product-depth k circuits as $(\Sigma\Pi)^k$ circuits, with the root gate being $+$ and the deepest layer of gates being $\times$. Since any constant depth algebraic circuit of depth k and size s can be converted to a formula of depth k and size s^{k+1} i.e. $\text{poly}(s)$, we will use the terms circuits and formulas interchangeably, without any loss in the final bounds we prove.
6. Let f and g be multivariate polynomials such that $g \mid f$. Then, the *multiplicity* or *factor multiplicity* of g in f is defined to be the greatest integer a such that g^a divides f.

2.3.1 Circuit/formula bit-complexity

Definition 2.3.1 (Bit-complexity of a circuit/formula)**.** *The* bit-complexity *of a circuit/formula* C*, denoted by* $\mathrm{bit}(C)$*, is defined as the sum of* $\mathrm{size}(C)$ *and the bit-complexities of all the scalars*[1] *present on edges or leaves. By default, any edge that does not have a scalar on it will be assigned the scalar* 1*.* ◊

Lemma 2.3.2 (Bit-complexity of evaluations of formulas)**.** *Let* C *be a formula of bit-complexity* s *computing a polynomial* $f(\overline{x})$*. If* $\overline{a} \in \mathbb{Q}^n$ *with each entry of* $\overline{a}$ *having bit-complexity* b*, then the bit-complexity of* $f(\overline{a})$ *is at most* $s \cdot b$*.*

(Proof deferred to Section 2.6)

2.3.2 Relevant subclasses of algebraic circuits

We briefly define subclasses of algebraic circuits that we would use often in this chapter.

Definition 2.3.3 (Power of low-degree polynomials)**.** *For a parameter* $d \in \mathbb{Z}_{\geqslant 0}$*, let* Deg_d *refer to the class of polynomials of degree at most* d*. We use* $(\mathrm{Deg}_d)^*$ *to denote the class of polynomials that are powers of polynomials of degree at most* d*.* ◊

Definition 2.3.4 ($\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$-formulas)**.** *We will use*

$$\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$$

to denote the subclass of algebraic formulas that compute expressions of the form

$$\sum_i f_i \cdot g_i^{e_i}$$

where each f_i *is a* $(\Sigma\Pi)^(k)$ *formula and each* g_i *is a polynomial of degree at most* d *and* e_i*'s are arbitrary positive integers. The size and bit-complexity of the above expression is defined as its size and bit-complexity when viewed as a general algebraic formula.* ◊

Observation 2.3.5. *Let* $\mathcal{C}$ *be the class of* $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ *formulas for fixed parameters* k *and* d*. Suppose* $P_1, \ldots, P_t$ *are polynomials computed by* $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ *formulas of size* s *and bit-complexity* b *each. Then,*

- $\sum_i P_i$ *is computable by an* $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ *formula of size at most* $t \cdot s$ *and bit-complexity at most* $O(t \cdot b)$*.*
- $\prod_i P_i$ *is computable by an* $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ *formula of size at most* $s^{O(t)}$ *and bit-complexity at most* $b^{O(t)}$*.*

(Proof deferred to Section 2.6.)

1 For a rational number $r = p/q$, its bit-complexity $\mathrm{bit}(r)$ is defined as $\log(\max(|p|, |q|))$

2.3.3 Standard preliminaries using interpolation

Lemma 2.3.6 (Univariate interpolation). *Let* $f(x) = f_0 + f_1 x + \cdots + f_d x^d$ *be a univariate polynomial of degree at most* d*. Then, for any* $0 \leqslant r \leqslant d$ *and there are*[2] *field constants* $\alpha_0, \ldots, \alpha_d$ *and* $\beta_{r0}, \ldots, \beta_{rd}$ *such that*

$$f_r = \beta_{r0} f(\alpha_0) + \cdots + \beta_{rd} f(\alpha_d).$$

Furthermore, the bit-complexity of all field constants is bounded by $\mathrm{poly}(d)$*.* □

Lemma 2.3.7 (Computing homogeneous components). *Let* $f \in \mathbb{Q}[\overline{x}]$ *be an* n*-variate degree* d *polynomial. Then, for an* $0 \leqslant i \leqslant d$*, there are field constants* $\alpha_0, \ldots, \alpha_d$ *and* β_{i0}, β_{id} *of bit-complexity* $\mathrm{poly}(d)$ *such that*

$$\mathrm{Hom}_i(f) = \beta_{i0} f(\alpha_0 \cdot \overline{x}) + \cdots + \beta_{id} f(\alpha_d \cdot \overline{x}).$$

In particular for $\mathcal{C} = (\Sigma\Pi)^k$ *or* $\Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$*, if* f *is computable by* $\mathcal{C}$*-formulas of size / bit-complexity at most* s *then* $\mathrm{Hom}_i(f)$ *is computable by* $\mathcal{C}$*-formulas of size / bit-complexity at most* $\mathrm{poly}(s, d)$*.*

Lemma 2.3.8 (Computing partial derivatives in one variable). *Let* $f \in \mathbb{Q}[\overline{x}]$ *be an* n*-variate degree* d *polynomial. Then, for an* $0 \leqslant r \leqslant d$*, there are field elements* α_i*'s and* β_{ij}*'s in* $\mathbb{Q}$ *of bit-complexity* $\mathrm{poly}(d)$ *such that*

$$\frac{\partial^r f}{\partial x_1^r} = \sum_{i=0}^{d} x_1^i \cdot (\beta_{i0} f(\alpha_0, x_2, \ldots, x_n) + \cdots + \beta_{id} f(\alpha_d, x_2, \ldots, x_n))$$

In particular for $\mathcal{C} = (\Sigma\Pi)^k$ *or* $\Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$*, if* f *is computable by* $\mathcal{C}$*-formulas of size / bit-complexity at most* s *then* $\frac{\partial^r f}{\partial x_1^r}$ *is computable by* $\mathcal{C}$*-formulas / bit-complexity of size at most* $O(s \cdot d^3)$*.*

Proof. We may consider the polynomial f as a univariate in x_1, and extract each coefficient of x_1^i using Lemma 2.3.6 and recombine them to get the appropriate partial derivative. That justifies the claimed expression.

As for the size, note that if $\mathcal{C}$ is $(\Sigma\Pi)^k$ or $\Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$, multiplying a size s formula by x_1^i, by using distributivity of the top addition gate, results in a $\mathcal{C}$-formula of size at most $s \cdot d$. Thus, the overall size of the above expression for the partial derivative is at most $O(s \cdot d^3)$. □

We will be making use of the following identity (which can be proved via appropriate interpolation as well).

Lemma 2.3.9 (Fischer's identity [Fis94; Ell69]). *If* $\mathbb{F}$ *is a field of characteristic zero or larger than* D*, then for any positive integers* $e_1, \ldots, e_n$

2 In fact, for any choice of distinct $\alpha_0, \ldots, \alpha_d$, there are appropriate $\beta_{r0}, \ldots, \beta_{rd}$ satisfying the equation. If the α_i's are chosen to have small bit-complexity, we can obtain a $\mathrm{poly}(d)$ bound on the bit-complexity of the associated β_{ri}'s.

with $\sum e_i = D$*, there are homogeneous linear forms* $L_1, \ldots, L_r$ *and field constants* $\alpha_1, \ldots, \alpha_r$ *of bit-complexity* $\mathrm{poly}(d, n)$ *such that*

$$x_1^{e_1} \cdots x_n^{e_n} = \sum_{i=1}^{r} \alpha_i L_i^D.$$

2.3.4 Polynomial identity testing

Lemma 2.3.10 (Polynomial Identity Lemma [Ore22; DL78; Sch80; Zip79])**.** *Let* $f \in \mathbb{Q}[\overline{x}]$ *be a non-zero* n *variate polynomial of degree at most* d*. Then, for every set* $S \subseteq \mathbb{Q}$*, the number of zeroes of* f *in the set* $S^n = S \times S \times \cdots \times S$ *is at most* $d|S|^{n-1}$*.*

Definition 2.3.11 (Low Hamming weight hitting set for low degree polynomials)**.** *Let* $n \geqslant d \geqslant 0$ *be integer parameters. Fix a set* $T_d \subseteq \mathbb{Q}$ *consisting of size* $(d+1)$*. The hitting set* $\mathcal{H}(d, n)$ *is defined as*

$$\mathcal{H}(d,n) = \left\{ (a_1, \ldots, a_n) \ : \ S \in \binom{[n]}{\leqslant d}, \ a_i \in T_d \text{ for all } i \in S \text{ and } a_j = 0 \text{ for all } j \notin S \right\}.$$

The size of the above set is at most $\binom{n}{\leqslant d} \cdot (d+1)^d = n^{O(d)}$*. Furthermore, choosing* T_d *to consist of elements of* $\mathbb{Q}$ *of bit-complexity* $\mathrm{poly}(d)$*, the bit-complexity of the set* $\mathcal{H}(d, n)$ *is bounded by* $n^{O(d)}$ *as well.* ◊

The following lemma is an easy consequence of Lemma 2.3.10 and will be crucial for parts of our proof. We also include a short proof sketch.

Lemma 2.3.12. *Let* $f \in \mathbb{Q}[\overline{x}]$ *be a non-zero* n *variate polynomial of degree at most* d*. Then, there exists a vector* $\overline{a} \in \mathcal{H}(d, n) \subseteq \mathbb{Q}^n$ *such that* $f(\overline{a}) \neq 0$*.*

(Proof deferred to Section 2.6.)

Theorem 2.3.13 (PIT for constant depth formulas (modification of Corollary 6 [LST25]))**.** *Let* $\epsilon > 0$ *be a real number and* $\mathbb{F}$ *be a field of characteristic* 0*. Let* C *be an algebraic formula of size and bit-complexity* $s \leqslant \mathrm{poly}(n)$*, depth* $k = o(\log\log\log n)$ *computing a polynomial on* n *variables, then there is a deterministic algorithm that can check whether the polynomial computed by* C *is identically zero or not in time* $(s^{O(k)} \cdot n)^{O_\epsilon((sD)^\epsilon)}$*.*

Theorem 2.3.14 (PIT for $\Sigma\left((\Sigma\Pi)^{(1)} \cdot (\mathrm{Deg}_d)^*\right)$ (Corollary 6.7, [For15]))**.** *Let* $t \geqslant 1$*. Then, the class* $\mathcal{C} = \Sigma\left((\Sigma\Pi)^{(1)} \cdot (\mathrm{Deg}_d)^*\right)$ *that computes polynomials of the form* $\sum_{i=1}^{s} f_i \cdot g_i^{d_i}$ *with each* f_i *being* s*-sparse and each* $\deg(g_i) \leqslant d$ *has a* $\mathrm{poly}(n, s, d\log s)$*-explicit hitting set of size* $\mathrm{poly}(s)^{O(d\log s)}$*.*

We will also crucially use the following lemma that gives an algorithm to obtain the coefficient vector of a polynomial from an algebraic formula computing it. In our setting, we invoke this algorithm only for low degree polynomials, and in that case, we can tolerate the runtime of this algorithm within our budget.

Lemma 2.3.15 (Interpolating a low degree multivariate polynomial)**.** *There is a deterministic algorithm that, when given a parameter* d *and an* n *variate algebraic formula* $C \in \mathbb{Q}[\overline{x}]$ *of size at most* s*, bit-complexity at most* b *and degree at most* d*, outputs the coefficient vector of the polynomial computed by* C*.*

The algorithm runs in time $\mathrm{poly}(s, b, n^d)$*.*

(Proof deferred to Section 2.6.)

2.3.5 Deterministic divisibility testing and PIT

Definition 2.3.16 (Pseudo-quotients)**.** *Let* $f, g \in \mathbb{Q}[\overline{x}]$ *be non-zero polynomials with* $g(\overline{0}) = \beta \neq 0$*. The* pseudo-quotient of f and g *is defined as*

$$\mathrm{Hom}_{\leqslant d_f - d_g}\left(\left(\frac{f(\overline{x})}{\beta}\right) \cdot (1 + \tilde{g} + \tilde{g}^2 + \cdots + \tilde{g}^{d_f - d_g})\right)$$

where $d_f = \deg(f)$*,* $d_g = \deg(g)$ *and* $\tilde{g} = 1 - \frac{g}{\beta}$*.*

More generally, if $\overline{\alpha} \in \mathbb{Q}^n$ *is such that* $g(\overline{\alpha}) \neq 0$*, the* pseudo-quotient of f and g translated by $\overline{\alpha}$ *is defined as the pseudo-quotient of* $f(\overline{x} + \overline{\alpha})$ *and* $g(\overline{x} + \overline{\alpha})$*.* ◊

The following lemma immediately follows from the above definition and Lemma 2.3.7.

Lemma 2.3.17 (Complexity of pseudo-quotients)**.** *Suppose* $k \geqslant 1$ *and* $f(\overline{x}) \in (\Sigma\Pi)^k$ *and* $g(\overline{x}) \in \mathrm{Deg}_d$ *of sizes at most* s_1, s_2 *respectively, and suppose* $g(\overline{0}) \neq 0$*. Then, the pseudo-quotient of* f, g *is computable by the* $\mathcal{C}$*-formulas of size at most* $\mathrm{poly}(s_1, s_2)$*, where* $\mathcal{C} = \Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$*.*

Theorem 2.3.18 (Divisibility testing to PIT [For15])**.** *Let* $f(\overline{x})$ *and* $g(\overline{x})$ *be non-zero* n *variate polynomials of degree* d_1 *and* d_2 *respectively over a field* $\mathbb{Q}$ *such that* $g(\mathbf{0}) = \beta \neq 0$*. Then,* g *divides* f *if and only if the polynomial* $R(\overline{x})$ *defined as*

$$R(\overline{x}) := f(\overline{x}) - g(\overline{x})Q(\overline{x})$$

is identically zero, where $Q(\overline{x})$ *is the pseudo-quotient of* f *and* g*.*

An immediate consequence of this theorem is the following corollary that takes into account the depth of an algebraic formula computing the polynomial $R(\overline{x})$ given above, assuming that f and g themselves can be computed by a low depth formula.

Corollary 2.3.19 (Divisibility testing to PIT for constant depth formulas [For15])**.** *Suppose* $f(\overline{x})$ *is a non-zero* n*-variate polynomial computed by a* $(\Sigma\Pi)^k$ *formula of size* s*, and suppose* $g(\overline{x})$ *is a polynomial of degree at most* d *with* $g(\overline{0}) = \beta \neq 0$*. Then, we can test if* g *divides* f *in time* $T(k, d, s')$ *where* $s' = \mathrm{poly}(s, d)$ *and* $T(k, d, s)$ *is the time required to test polynomial identities of the size* s *expressions of the form*

$$\Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right).$$

(Proof deferred to Section 2.6.)

Theorem 2.3.20 ([For15]). *Let $\mathbb{F}$ be any sufficiently large field. Then, there is a deterministic algorithm that takes an input two polynomials f and g and parameters d, D, n, s, where f is an n-variate polynomial of degree at most D and sparsity s; g is an n variate polynomial of degree d, and outputs whether g divides f or not in time $\exp(O(d \log^2 snDd))$.*

2.3.6 Resultants

Definition 2.3.21 (The Resultant). *Let $\mathcal{R}$ be a commutative ring. Given polynomials g and h in $\mathcal{R}[y]$, where:*

$$g(y) = g_0 + \cdots + y^d \cdot g_d$$
$$h(y) = h_0 + y \cdot h_1 + \cdots + y^D \cdot h_D$$

with g_d and $h_D \neq 0$ the Resultant *of g and h, denoted by $\mathrm{Res}_y(g, h)$, is the determinant of the $(D + d) \times (D + d)$ Sylvester matrix Γ of g and h, given by:*

$$\Gamma = \begin{bmatrix} h_0 & h_1 & \dots & & h_D & & \\ & \ddots & \ddots & & \ddots & \ddots & \\ & & h_0 & h_1 & & \dots & h_D \\ g_0 & \dots & & g_d & & & \\ & g_0 & \dots & & g_d & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_0 & \dots & & g_d \end{bmatrix}$$

$\Diamond$

Lemma 2.3.22 (Resultant and gcd (Corollary 6.20 [GG13])). *Let $\mathcal{R}$ be a unique factorisation domain and $g, h \in \mathcal{R}[y]$ be non-zero polynomials. Then:*

$$\deg_y(\gcd(g, h)) > 0 \iff \mathrm{Res}_y(g, h) = 0$$

where $\gcd(g, h) \in \mathcal{R}[y]$ and $\mathrm{Res}_y(g, h) \in \mathcal{R}$.

In this chapter, $\mathcal{R}$ will be $\mathbb{Q}[\overline{x}]$ (which is a unique factorisation domain), and $\mathrm{Res}_y(g, h)$ will denote the resultant of $g, h \in \mathbb{Q}[\overline{x}][y]$ when considered as polynomials in $\mathcal{R}[y]$. We might also occasionally refer to it as the *y-resultant* of g and h. For more details about the resultant as well as a proof of the above lemma, we refer the reader to von zur Gathen and Gerhard's book on computer algebra (Chapter 6, [GG13]). We mention a simple observation from the above definition that would be useful for this chapter.

Observation 2.3.23 (Resultant under substitutions). *Suppose $g(\overline{x}, y) = g_0(\overline{x}) + g_1(\overline{x})y + \cdots g_d(\overline{x})y^d$ and $h(\overline{x}, y) = h_0(\overline{x}) + h_1(\overline{x})y + \cdots + h_D(\overline{x})y^D$*

with $g_d, h_D \neq 0$. *Then, for any* $\overline{a} \in \mathbb{Q}^{|\overline{x}|}$ *that ensures* $g_d(\overline{a}), h_D(\overline{a}) \neq 0$, *we have*

$$(\mathrm{Res}_y(g,h))(\overline{a}) = \mathrm{Res}_y(g(\overline{a},y), h(\overline{a},y)). \qquad \square$$

2.3.7 Hensel Lifting

Now we will state the definition of a *lift* and the main lemma for Hensel lifting. For more details, one can look up some of the cited papers or the standard references in computational algebra [KSS15; ST20; GG13; Sud98].

Definition 2.3.24 (Hensel lifts). *Let* $\mathcal{I} \subseteq \mathbb{Q}[\overline{x}, y]$ *be an ideal. Let* $f, g, h, u, v \in \mathbb{Q}[\overline{x}, y]$ *such that* $f \equiv gh \bmod \mathcal{I}$ *and* $ug + vh \equiv 1 \bmod \mathcal{I}$. *Then, we call* $g', h' \in \mathbb{Q}[\overline{x}, y]$ *a* lift *of* g *and* h *if:*

1. $f \equiv g'h' \bmod \mathcal{I}^2$,
2. $g' \equiv g \bmod \mathcal{I}$ *and* $h' \equiv h \bmod \mathcal{I}$, *and*
3. $\exists u', v' \in \mathbb{Q}[\overline{x}, y]$ *s.t* $u'g' + v'h' \equiv 1 \bmod \mathcal{I}^2$. $\Diamond$

For the rest of the section, we define $\mathcal{I}$ to be the ideal $\langle x_1, \ldots, x_n \rangle$ and $\mathcal{I}_k := \mathcal{I}^{2^k}$.

Lemma 2.3.25 (Iterated monic Hensel lifting (Lemma 3.4 [KSS15])). *Suppose we're given* $f, g, h \in \mathbb{Q}[\overline{x}, y]$ *such that* $f = gh$, g *is monic in* y *and* $\gcd(g,h) = 1$. *We are also given* $g_0, h_0, u_0, v_0 \in \mathbb{Q}[\overline{x}, y]$ *such that* $g_0 \equiv g \bmod \mathcal{I}$, $h_0 \equiv h \bmod \mathcal{I}$ *and* $u_0 g_0 + v_0 h_0 \equiv 1 \bmod \mathcal{I}$. *Then, for all* $k \in \mathbb{N}, k \geqslant 1$, *we can compute* $g_k, h_k, u_k, v_k \in \mathbb{Q}[\overline{x}, y]$, *with each* g_k *being monic, such that the following conditions hold:*

1. *The pair* g_k, h_k *is a lift of* g_{k-1}, h_{k-1}, *with* $u_k g_k + v_k h_k \equiv 1 \bmod \mathcal{I}_k$; *in particular,* $f \equiv g_k h_k \bmod \mathcal{I}_k$
2. $g_k \equiv g \bmod \mathcal{I}_k$ *and* $h_k \equiv h \bmod \mathcal{I}_k$

Moreover, for each k, g_k *and* h_k *are unique polynomials modulo* $\mathcal{I}_k$ *satisfying the above conditions when the* g_k*s are monic. For each* k, *we will call* g_k, h_k *the* k-th iterated lift of g_0, h_0.

If $\deg_{\overline{x}}(g) = d$, we can choose an integer k^* such that $d < 2^{k^*} \leqslant 2d$ and use the above Lemma to get $g_{k^*} \equiv g \bmod \mathcal{I}_{k^*}$, which means we can truncate g_{k^*} to degree d and retrieve g. The next lemma tells us that this can be done with reasonable bounds on the parameters of the underlying circuits.

Lemma 2.3.26 (Small circuit for Hensel lifting (Lemma 3.6 [KSS15])). *Let* f *be a degree* D *polynomial in* $\mathbb{Q}[\overline{x}, y]$, *computable by a* $(\Sigma\Pi)^{k+1}$ *formula of size and bit-complexity* s, *with a factorisation* $f = gh$ *such that* $\gcd(g,h) = 1$ *and* g *is monic. Let* $g_0 = g \bmod \mathcal{I}$ *and* $h_0 = h \bmod \mathcal{I}$ *be univariates in* $\mathbb{Q}[y]$ *with* $\gcd(g_0, h_0) = 1$.

Then, there is a formula C_g, C_h *of size and bit complexity* $(sDk)^{O(k \log D)}$ *that computes the* k^{th} *iterated lift* g_k, h_k *of* g_0, h_0*, where* g_k *is monic. More generally, if the total degree of* g_k *is at most* d*, then the size and bit complexity of the formula for* g_k *are at most* $(sDk)^{O(\log d)}$*.*

Moreover, there is a deterministic algorithm, that when given the formulas for f *and* g_0, h_0 *and integer* k *as input, outputs the formulas for* g_k *and* h_k *in time* $(sDk)^{O(k \log D)}$ *(resp.* $(sDk)^{O(\log d)}$ *if* g_k *has total degree* d*).*

(Proof sketch deferred to Section 2.6.)

2.3.8 Results on polynomial factorisation

We rely on the following two fundamental results on polynomial factorisation for our results. The first theorem is a classical algorithm of Lenstra, Lenstra and Lovász for factoring univariate polynomials over the field of rational numbers.

Theorem 2.3.27 (Factorising polynomials with rational coefficients [LLL82; GG13])**.** *Let* $f \in \mathbb{Q}[x]$ *be a monic polynomial of degree* d*. Then there is a deterministic algorithm computing all the irreducible factors of* f *that runs in time* $\mathrm{poly}(d, t)$*, where* t *is the maximum bit-complexity of the coefficients of* f*.*

The second result we need is an easy consequence of the results of Kopparty, Saraf and Shpilka [KSS15]. They showed that an efficient deterministic algorithm for PIT for algebraic circuits implies an efficient deterministic algorithm for polynomial factorisation. The formal statement below essentially invokes this for constant degree polynomials. In this case, the PIT instances also happen to be of constant degree and hence can be easily solved in time that is polynomial in the length of the coefficient vector of these polynomials.

Theorem 2.3.28 ([KSS15])**.** *There is a deterministic algorithm that when given as input the coefficient vector of an* n *variate polynomial* $f(\overline{x}) \in \mathbb{Q}[\overline{x}]$ *of total degree* d*, runs in time* $n^{O(d^2)}$ *and decides if* f *is irreducible or not.*

2.4 COMPUTING CANDIDATE LOW-DEGREE FACTORS OF MULTIPLICITY ONE

We first present the algorithm for computing candidate low-degree factors of multiplicity one in Algorithm 1 below. In the next section, we use this as a subroutine in Algorithm 2 to compute factors of all multiplicity and also eliminate those candidates that were not actual factors.

Algorithm 1: Computing candidate degree d factors of factor-multiplicity one

Input : A $(\Sigma\Pi)^k$-formula of size s, bit-complexity t, degree D computing a polynomial $f(\overline{x})$.

Output: A list of polynomials of degree at most d, that include all factors of f with degree at most d and multiplicity 1.

Set the output list $L = \emptyset$.

Compute hitting-set $H_1 = \mathcal{H}(d, n)$ (as defined in Definition 2.3.11).

Compute hitting-set H_2 for the class $\mathcal{C} = \Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$ of size $(sD)^{O(d)}$.

for $\overline{\alpha}, \overline{\beta} \in H_1$ *and* $\overline{\delta} \in H_2$ **do**

  Define $F(\overline{x}, y) = f(\overline{x} + \overline{\alpha} \cdot y + \overline{\beta} + \overline{\delta}) = f(x_1 + \alpha_1 y + \beta_1 + \delta_1, \ldots, x_n + \alpha_n y + \beta_1 + \delta_n)$

  Using interpolation on the formula for $F(\overline{x}, y)$ (via Lemma 2.3.6), compute $F(\overline{0}, y)$ as a sum of monomials.

  Factorise the polynomial $F(\overline{0}, y)$ into irreducible factors as

$$F(\overline{0}, y) = \sigma \cdot F_1^{e_1} \cdots F_r^{e_r}.$$

  where $0 \neq \sigma \in \mathbb{Q}$ and each F_r is monic in y.

  for $T \subseteq [r]$ *of size at most* d **do**

    Define $g_0 = \prod_{i \in T} F_i^{e_i}$ and $h_0 = \sigma \cdot \prod_{i \notin T} F_i^{e_i}$, interpretted as polynomials in $\mathbb{Q}[\overline{x}, y]$.

    if $\deg(g_0) > d$ **then**

      Continue to the next choice of T in the current loop.

    Compute polynomials u_0, v_0 such that $u_0 g_0 + v_0 h_0 = 1$.

    Use Hensel-Lifting (Lemma 2.3.26) to lift the factorisation $F(\overline{x}, y) = g_0(\overline{x}, y) \cdot h_0(\overline{x}, y) \bmod I$, where $I = \langle \overline{x} \rangle$, to obtain algebraic circuits for g_ℓ, h_ℓ satisfying

$$F(\overline{x}, y) = g_\ell(\overline{x}, y) \cdot h_\ell(\overline{x}, y) \bmod I^{2^\ell}$$

    with g_ℓ being monic and $d < 2^\ell < 2d$.

    Using interpolation on the circuit for g_ℓ (via Lemma 2.3.15), compute g_ℓ as a sum of monomials.

    Add $\tilde{g} = g_\ell(\overline{x} - \overline{\delta} - \overline{\beta}, 0)$ to L.

return L

Before we discuss the proof of correctness and running time of Algorithm 1, we state two simple observations that we use in the analysis. We defer the proofs of these observations to the end of the section.

Observation 2.4.1 (Size growth under a translation of low Hamming weight)**.** *Let $k > 0$ be a parameter. Let $f(\overline{x})$ be an n-variate polynomial*

of degree at most D *with* $(\Sigma\Pi)^k$*-size at most* s*. If* $\overline{\alpha}, \overline{\beta} \in \mathcal{H}(d, n)$*, the polynomial* $\tilde{f}(\overline{x}, y) = f(\overline{x} + y\overline{\alpha} + \overline{\beta})$ *has* $(\Sigma\Pi)^k$*-size at most* $s \cdot D^{O(d)}$*.*

Lemma 2.4.2. *Let* $f(\overline{x})$ *be an* n*-variate polynomial computed by a* $(\Sigma\Pi)^k$ *formula of size* s*, and let* $g(\overline{x})$ *be an* n*-variate degree* d *polynomial with* $g(\overline{0}) \neq 0$*. Let* $Q(\overline{x})$ *be the pseudo-quotient of* f *and* g*. Then, for any variable* $y \in \overline{x}$*, the polynomial* $\mathrm{Res}_y(Q, g)$ *is computable by a* $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ *formula of size at most* $s^{O(d)}$*.*

2.4.1 Proof of correctness of the Algorithm 1

Lemma 2.4.3 (Correctness of Algorithm 1)**.** *For every input polynomial* f *computed by* $(\Sigma\Pi)^k$ *formulas of size* s*, bit-complexity* t*, degree* D *and any factor* g *of degree at most* d *with* $g \mid f$ *and* $g^2 \nmid f$*, the polynomial* g *is included in the output list of Algorithm 1 on input* f*.*

Proof. Algorithm 1 outputs a list of candidate factors; we would like to prove that every factor of f with degree $\leqslant d$ and factor-multiplicity one will be contained in this list.

Fix any specific factor g of f, with $\deg(g) = d' \leqslant d$ and factor-multiplicity one, which ensures that $\gcd(g, f/g) = 1$. The coefficient of $y^{d'}$ in $g'(\overline{x}, y) := g(\overline{x} + y\overline{\alpha} + \overline{\beta})$ is the evaluation of $\mathrm{Hom}_{d'}(g)$ at $\vec{\alpha}$ and the constant term of $g'(\overline{x}, y)$ is $g'(\overline{0}, 0) = g(\overline{\beta})$. Thus, there is some $\overline{\alpha}, \overline{\beta} \in H_1$ suth that $\mathrm{Hom}_{d'}(g)(\overline{\alpha}) \neq 0$ and $g(\overline{\beta}) \neq 0$. Fix this choice of $\overline{\alpha}, \overline{\beta}$. We then have that $g'(\overline{x}, y)$ is monic in y and has non-zero constant term.

With the above properties, the pseudo-quotient h' of $f'(\overline{x}, y) := f(\overline{x} + y\overline{\alpha} + \overline{\beta})$ and $g'(\overline{x}, y)$ is well-defined and is a polynomial in $\Sigma\left((\Sigma\Pi)^k \cdot (\mathrm{Deg}_d)^*\right)$ (by Lemma 2.3.17) of size $\mathrm{poly}(s, D, d) \leqslant \mathrm{poly}(sD)$. By Lemma 2.4.2, $\mathrm{Res}_y(g', h') \in \mathbb{Q}[\overline{x}]$ is non-zero polynomial computable by $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ formulas of size $(sD)^{O(d)}$.

Let $\deg_y(h') = r$ and $h'(\overline{x}, y) = h'_0(\overline{x}) + \cdots + h'_r(\overline{x})y^r$. Since h' is computable by size $(sD)^{O(d)}$ formula from $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$, so is the leading term $h'_r(\overline{x})$ by Lemma 2.3.6. Therefore by Observation 2.3.5, the polynomial $\Gamma(\overline{x}) = \mathrm{Res}_y(g', h') \cdot h_r(\overline{x})$ is also computable by $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ formulas of size $s' = (sD)^{O(d)}$. Since H_2 is a hitting set for $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ formulas of size s', fix a $\overline{\delta} \in H_2$ such that $\Gamma(\overline{\delta}) \neq 0$. By Observation 2.3.23 and Lemma 2.3.22, we have that $g'(\overline{\delta}, y)$ and $h'(\overline{\delta}, y)$ are coprime polynomials. Thus, if $g''(\overline{x}, y) = g'(\overline{x} + \overline{\delta}, y)$ and $h''(\overline{x}, y) = h'(\overline{x} + \overline{\delta}, y)$, we have

$$\begin{aligned} & f(\overline{x} + \overline{\alpha}y + \overline{\beta} + \overline{\delta}) = g''(\overline{x}, y) \cdot h''(\overline{x}, y) \\ \implies & f(\overline{\alpha}y + \overline{\beta} + \overline{\delta}) = g''(\overline{0}, y) \cdot h''(\overline{0}, y) \\ & \qquad\qquad \text{with } \gcd(g''(\overline{0}, y), h''(\overline{0}, y)) = 1. \end{aligned}$$

Line 7 thus factorises the univariate polynomial $f(\overline{\alpha}y + \overline{\beta} + \overline{\delta})$ and one of the sets T in Line 8 must correspond to $g_0(y)$ chosen in Line 9

to satisfy $g_0(y) = g''(\overline{0}, y)$ and $h_0(y) = h''(\overline{0}, y)$. Thus, we have a factorisation of the form

$$\begin{aligned} f(\overline{\alpha}y + \overline{\beta} + \overline{\delta}) &= g''(\overline{0}, y) \cdot h''(\overline{0}, y) = g_0 \cdot h_0 \\ \implies f(\overline{x} + \overline{\alpha}y + \overline{\beta} + \overline{\delta}) &= g_0 \cdot h_0 \bmod \mathcal{I}, \quad \text{where } \mathcal{I} = \langle \overline{x} \rangle. \end{aligned}$$

We are therefore set-up to apply Hensel Lifting (Lemma 2.3.25) to obtain g_ℓ, h_ℓ such that g_ℓ is monic in y and

$$f(\overline{x} + \overline{\alpha}y + \overline{\beta} + \overline{\delta}) = g_\ell(\overline{x}, y) \cdot h_\ell(\overline{x}, y) \bmod \mathcal{I}^{2^\ell}.$$

From the uniqueness of Hensel Lifting (which is guaranteed by Lemma 2.3.25), we must have that $g_\ell(\overline{x}, y) = g''(\overline{x}, y) = g(\overline{x} + \overline{\alpha}y + \overline{\beta} + \overline{\delta})$. Thus, for this choice of $\overline{\alpha}, \overline{\beta}, \overline{\delta}$ and T, we would include $g(\overline{x}) = g''(\overline{x} - \overline{\beta} - \overline{\delta}, 0)$ in the set of candidate factors in Line 15.

Finally, since the lift also ensures that there exist u_ℓ and v_ℓ such that $u_\ell g_\ell + v_\ell h_\ell = 1 \bmod \mathcal{I}^{2^\ell}$, we also have that $g_\ell^2 \nmid f$. □

2.4.2 Running time analysis

We now bound the time complexity of the algorithm.

Lemma 2.4.4 (Running time of Algorithm 1)**.** *Let $\epsilon > 0, d, k \in \mathbb{N}$ be an arbitrary constants and let $f \in \mathbb{Q}[\overline{x}]$ be a polynomial computable by a $(\Sigma\Pi)^k$ formula C of size s, degree at most D and bit-complexity t. Then, on input C, Algorithm 1 terminates in time at most $(sD)^{O(d(sD)^{\epsilon d})}$.*

Moreover, if $k = 1$, i.e. f has sparsity at most s, then Algorithm 1 terminates in time at most $(sDt)^{(\mathrm{poly}(d) \log sDt)}$.

Proof. Let $T_k^{(1)}(s, d)$ be the time-complexity to output the hitting set H_1 in Line 2 and $T_k^{(2)}(s, D, d)$ be the time-complexity to output the hitting set H_2 in Line 3.

From Definition 2.3.11, we immediately have that $T_k^{(1)}(s, d) \leqslant s^{O(d)}$. As for $T_k^{(2)}(s, D, d)$, in the case of $k = 1$, Theorem 2.3.14 shows that $T_k^{(2)}(s, D, d) \leqslant (sD)^{(\mathrm{poly}(d) \log sD)}$. For k satisfying $2 \leqslant k = o(\log\log\log s)$, then Theorem 2.3.13 shows that $T_k^{(2)}(s, D, d) \leqslant (sD)^{O(d(sD)^{\epsilon d})}$ for any constant $\epsilon > 0$.

Using Lemma 2.3.6, we get that Line 6 takes $\mathrm{poly}(s, D, t)$-time. Now, each of the coefficients of $F(\overline{0}, y)$ has bit-complexity at most $\mathrm{poly}(s, D, t)$. Thus, from Theorem 2.3.27, we get that $F(\overline{0}, y)$ can be factorised into its irreducible factors in time at most $\mathrm{poly}(s, D, t)$.

There are at most D^d choices for the set T in Line 8. For each such choice, Line 12 computes formulas of size $\mathrm{poly}(s, D, t)$ for g_0, h_0, u_0, v_0 in time $\mathrm{poly}(s, D, t)$. By Lemma 2.3.26, we have that Line 13 takes time $(sDt)^{O(\log d)}$ to compute a formula of the same size and bit-complexity for g_ℓ. From Lemma 2.3.15, we get that we can obtain the coefficient vector of g_ℓ in time at most $(sDt)^{O(d \log d)}$.

Therefore, the overall running time of Algorithm 1 is at most

$$T_k^{(1)}(s,d) \cdot T_k^{(2)}(s,D,d) \cdot D^d \cdot \mathrm{poly}(s,D,t) \cdot (sDt)^{O(d \log d)}.$$

Plugging in the estimates for $T_k^{(1)}(s,d)$, $T_k^{(2)}(s,D,d)$, we get the overall bound of $(sD)^{O(d(sD)^{\varepsilon d})}$ for $k > 1$, which is essentially dominated by $T_k^{(2)}(s,D,d)$.

When f has sparsity s, then as discussed in the proof, $T_k^{(2)}(s,d)$ is at most $(sD)^{(\mathrm{poly}(d) \log sD)}$. Plugging this back in the above expression, we get that the running time is at most $(sDt)^{(\mathrm{poly}(d) \log sDd)}$. □

2.4.3 Proof of structural lemmas

In this subsection, we include the proofs of Observation 2.4.1 and Lemma 2.4.2. This completes the analysis of Algorithm 1.

Proof of Observation 2.4.1. By definition of $\mathcal{H}(d,n)$ (Definition 2.3.11), the transformation $\overline{x} \mapsto \overline{x} + y\overline{\alpha} + \overline{\beta}$ takes a monomial $\prod_{i\in[n]} x_i^{e_i}$ to $\left(\prod_{i\in T}(x_i + \alpha_i y + \beta_i)^{e_i}\right) \cdot \left(\prod_{i\in[n]\setminus T} x_i^{e_i}\right)$, for some $T \subseteq [n]$ s.t. $|T| = d$. If we expand $\prod_{i\in T}(x_i + \alpha_i y + \beta_i)^{e_i}$ into a sum of monomials, we will get at most $D^{O(d)}$ monomials (when $\sum_i e_i \leqslant D$). Expanding each $\prod_{i\in[n]}(x_i + \alpha_i y + \beta_i)^{e_i}$ at the bottom layer into a sum of monomials this way, we get the required $(\Sigma\Pi)^k$ formula with size at most $s \cdot D^{O(d)}$. □

Proof of Lemma 2.4.2. Let $\overline{x}' = \overline{x} \setminus \{y\}$ and let $\mathcal{C}$ be the class $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$. Let us assume that $\deg_y(Q) = D \leqslant s$ and $\deg_y(g) = d$. By Lemma 2.3.17, we have that $Q(\overline{x})$ is computable by a $\mathcal{C}$-formula of size at most $\mathrm{poly}(s,d)$. Let us consider the $(D+d) \times (D+d)$ Sylvester matrix Γ of Q and g with respect to the variable y whose determinant is $\mathrm{Res}_y(Q,g)$.

$$Q(\overline{x}) = Q_0(\overline{x}') + y \cdot Q_1(\overline{x}') + \cdots + y^D \cdot Q_D(\overline{x}')$$
$$g(\overline{x}) = g_0(\overline{x}') + \cdots + y^d \cdot g_d(\overline{x}')$$
$$\Gamma = \begin{bmatrix} Q_0 & Q_1 & \cdots & & Q_D & & \\ & \ddots & \ddots & & \ddots & \ddots & \\ & & Q_0 & Q_1 & & \cdots & Q_D \\ g_0 & \cdots & & g_d & & & \\ & g_0 & \cdots & & g_d & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_0 & \cdots & & g_d \end{bmatrix}$$

Note that, by Lemma 2.3.6, each of the Q_i's are computed by a $\mathcal{C}$-formula of size $\mathrm{poly}(s, D)$ and each g_i is a polynomial of degree at most d.

For a subset S of rows and T of columns, we will use $\Gamma(S, T)$ to refer to the submatrix restricted to the rows in S and columns in T, and let $\mathrm{Top} = \{1, \ldots, d\}$ and $\mathrm{Bot} = \{d+1, \ldots, d+D\}$. The determinant of Γ can then be expressed as

$$\det(\Gamma) = \mathrm{Res}_y(Q, g) = \sum_{T \in \binom{[D+d]}{d}} \det(\Gamma(\mathrm{Top}, T)) \cdot \det(\Gamma(\mathrm{Bot}, \overline{T}))$$

For every choice of T, the polynomial $\det(\Gamma(\mathrm{Top}, T))$ is the determinant of a $d \times d$ matrix each of whose entries are computable by $s' = \mathrm{poly}(s, D)$ sized $\mathcal{C}$-formulas. Therefore, using Observation 2.3.5, the polynomial $\det(\Gamma(\mathrm{Top}, T))$ is computable by $\mathcal{C}$-formulas of size at most $(sD)^{O(d)}$.

The polynomial $\det(\Gamma(\mathrm{Bot}, \overline{T}))$ is a degree D polynomial combination of $g_0, \ldots, g_d$ and can therefore be expressed as

$$\begin{aligned}\det(\Gamma(\mathrm{Bot}, \overline{T})) &= \sum_{i=1}^{D^{d+1}} a_i \cdot g_0^{e_{i,0}} \cdots g_d^{e_{i,d}} \\ &= \sum_{i=1}^{D^{d+1}} a_i \cdot \left(\sum_{j=1}^{D^{O(d)}} b_{ij} \cdot f_{ij}^{e_{ij}} \right) \quad \text{(using Lemma 2.3.9).}\end{aligned}$$

for some polynomials f_j of degree at most d. Thus, using Observation 2.3.5 again, we have that $\mathrm{Res}_y(Q, g)$ is computable by $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ formulas of size at most $D^{O(d)} \cdot (sD)^{O(d)} = (sD)^{O(d)}$. □

2.5 COMPUTING FACTORS OF ALL MULTIPLICITY

The following lemma essentially shows that the multiplicity of any factor g of a given polynomial f can be reduced by working with appropriate partial derivatives of f, with respect to variables that are present in g. This naturally yields an algorithm that uses Algorithm 1 as a subroutine, and computes all irreducible factors of f.

Lemma 2.5.1 (Reducing factor multiplicity)**.** *Let $f(\overline{x}), g(\overline{x}) \in \mathbb{Q}[\overline{x}]$ be non-zero polynomials and let $x \in \overline{x}$ be such that $\partial_x(g) \neq 0$ and g is square-free. Then, the factor-multiplicity of g in f (i.e. the integer a satisfying $g^a \mid f$ and $g^{a+1} \nmid f$) is also the smallest non-negative integer a such that $g \nmid \frac{\partial^a f}{\partial x^a}$.*

Proof. If the factor-multiplicity of g in f is zero, i.e. $g \nmid f$, then claim is clearly true. Thus let us assume that the factor-multiplicity of g in f is $a \geqslant 1$. It suffices to show that the factor-multiplicity of g in $\partial_x(f)$ is exactly $a - 1$.

Suppose $f = g^a \cdot h$ where $\gcd(g, h) = 1$. Then,

$$\partial_x f = \partial_x(g^a) \cdot h + g^a \cdot \partial_x(h) = g^{a-1} \cdot (a \cdot \partial_x(g) \cdot h + g \cdot \partial_x(h)).$$

Hence, we have that the factor-multiplicity of g in $\partial_x(f)$ is at least $(a-1)$.

On the other hand, we have that $\partial_x(g) \neq 0$ and g is square-free and hence $\gcd(g, \partial_x(g)) = 1$. Therefore

$$\gcd(g, a \cdot g \cdot \partial_x(h) + h \cdot \partial_x(g)) = \gcd(g, h \cdot \partial_x(g)) = \gcd(g, h) = 1$$

and hence $g^a \nmid \partial_x(f)$ and therefore the factor-multiplicity of f □

We are now ready to describe the algorithm.

Algorithm 2: Computing list of all degree d irreducible factors and their multiplicities

Input : A $(\Sigma\Pi)^k$-formula of size s, bit-complexity t, degree D computing a polynomial $f(\overline{x})$.

Output: A list of all irreducible factors f of degree at most d and their multiplicities.

1 Set the output list $L = \emptyset$.
2 Set the intermediate candidates list $L' = \emptyset$.
3 Compute hitting-set $H_1 = \mathcal{H}(d, n)$ (as defined in Definition 2.3.11).
4 **for** $\overline{\alpha} \in H_1$ **do**
5 Define $F(\overline{x}, y) = f(\overline{x} + \overline{\alpha} \cdot y) = f(x_1 + \alpha_1 y, \ldots, x_n + \alpha_n y)$
6 **for** $i = 0, 1, \ldots, \deg(F)$ **do**
7 Define $\tilde{F}(\overline{x}, y) = \frac{\partial^i F}{\partial y^i}$.
8 Compute the list $\tilde{L}$ of all candidate degree d multiplicity-one factors of $\tilde{F}(\overline{x}, y)$ using Algorithm 1.
9 **foreach** $\tilde{g}(\overline{x}, y) \in \tilde{L}$ **do**
10 Add $g(\overline{x}) := \tilde{g}(\overline{x}, 0)$ to L'.
11 **for** $g \in L'$ **do**
12 **if** g is not irreducible **then** skip to the next iteration.
13 Let x be a variable that g depends on, so that $\partial_x(g) \neq 0$.
14 Find the smallest non-negative integer e such that $g \nmid \frac{\partial^e f}{\partial x^e}$.
15 **if** $e > 1$ **then** add (g, e) to the list L.
16 **return** L

Lemma 2.5.2 (Correctness of Algorithm 2)**.** *For every input polynomial f computed by a $(\Sigma\Pi)^k$ formula of size s, degree D, bit-complexity t and $d \in \mathbb{N}$, the list L output by Algorithm 2 is precisely the list of all irreducible factors of f of degree at most d (up to scalar multiplication) along with their multiplicities in f.*

Proof. From Line 11 to Line 15, and Lemma 2.5.1, it is clear that any (g, e) in the output list ensures that g is an irreducible polynomial, $g^e \mid f$ and $g^{e+1} \nmid f$. Thus, it suffices to show that for every irreducible polynomial g such that $\deg(g) \leqslant d$ and $g \mid f$, some non-zero scalar

multiple of g is under consideration in the list L'. Fix any such irreducible factor g of degree at most $r \leqslant d$ and let its factor-multiplicity be e

By Lemma 2.3.12, there is some $\overline{\alpha} \in H_1$ such that $\mathrm{Hom}_r(g)(\overline{\alpha}) \neq 0$, where r is the total degree of g. Thus, for this choice of $\overline{\alpha}$, we have that $g'(\overline{x}, y) = g(\overline{x} + y\overline{\alpha})$ is a factor of $F(\overline{x}, y) = f(\overline{x} + y\overline{\alpha})$ and g' is monic in y and has factor-multiplicity e. By Lemma 2.5.1, we have that g' has factor-multiplicity one in $\tilde{F}(\overline{x}, y) := \frac{\partial^{e-1} F}{\partial y^{e-1}}$. Thus, by the correctness of Algorithm 1 (Lemma 2.4.3), a non-zero multiple of the polynomial $g'(\overline{x}, y)$ must be included in the list $\tilde{L}$ in Line 8. Therefore, a non-zero multiple of $g(\overline{x}) = g'(\overline{x}, 0)$ will be added to L' in Line 10. □

Lemma 2.5.3 (Running time of Algorithm 2)**.** *Let $\epsilon > 0, k, d \in \mathbb{N}$ be arbitrary constants. Let $f \in \mathbb{Q}[\overline{x}]$ be a polynomial computable by a $(\Sigma\Pi)^k$ formula C of size s, degree at most D and bit-complexity t. Then, on input C and $d \in \mathbb{N}$, Algorithm 1 terminates in time at most $(sDt)^{O(d(sDt)^{\epsilon d})}$.*

Moreover, if $k = 1$, i.e. f has sparsity at most s, then Algorithm 1 terminates in time at most $(snDt)^{O(\mathrm{poly}(d) \cdot \log snDt)}$.

Proof. From Definition 2.3.11, we have the size of the set H_1 is $n^{O(d)}$. The time complexity of computing a formula for F from the given formula for f is at most $O(sD)$. From Lemma 2.3.8, we have that $(\Sigma\Pi)^{k+1}$ formulas for all the y derivatives of F can be computed in time at most $\mathrm{poly}(s, D, t)$, which is also a bound on the bit-complexity and the size of these formulas. Algorithm 1 is invoked at most D times. The total time taken to construct the list L' is at most $D \cdot T_1$, where T_1 is the time taken by Algorithm 1 on inputs with formula size and bit-complexity $\mathrm{poly}(s, D, t)$, and degree parameter d. $D \cdot T_1$ is also an upper bound on the size of the list of candidate factors L'.

Now, for each $g \in L'$, from Theorem 2.3.28, we have that the irreducibility test in Line 12 takes at most $(sDt)^{O(d^2)}$ time. There are at most D instances of divisibility test performed to determine the exact multiplicity in f of each $g \in L'$. This requires computing the corresponding derivatives, which as discussed in the previous paragraph, takes time $\mathrm{poly}(s, D, t)$ and outputs a formula of size and bit-complexity $\mathrm{poly}(s, D, t)$ for the derivatives, and then doing a divisibility test, the time complexity of which we denote by T_2.

Therefore, the total time taken by the algorithm is at most $(n^{O(d)} \cdot \mathrm{poly}(s, D, t) \cdot D \cdot T_1) + (D \cdot T_1 \cdot (sDt)^{O(d^2)} \cdot \mathrm{poly}(s, D, t) \cdot T_2)$.

Now, if f is s sparse, i.e. $k = 1$, then from Definition 2.3.11, we have that every vector in H_1 has at most d non-zero coordinates. Thus, from Observation 2.4.1, for every $\overline{\alpha} \in H_1$, $F(\overline{x}, y) = f(\overline{x} + \overline{\alpha} \cdot y)$ has sparsity and bit-complexity at most $s' \leqslant s \cdot D^d$. Note that the derivatives of arbitrary order of F with respect to any variable also have the same bound on their sparsity and bit-complexity of coefficients. Thus, in this case, from Lemma 2.4.4, $T_1 \leqslant (sDt)^{\mathrm{poly}(d) \log sDt}$. From Theorem 2.3.20,

we have that $T_2 \leqslant (snD)^{O(d \log^2 snD)}$. Therefore, the overall running time of the algorithm is at most $(snDt)^{O(\mathrm{poly}(d) \cdot \log snDt)}$.

On the other hand, if $k > 1$, then from Lemma 2.4.4, $T_1 \leqslant (sDt)^{O(d(sDt)^{\epsilon d})}$. To bound T_2 in this case, we note from Corollary 2.3.19, this divisibility testing instances reduce to PIT instances for $(\Sigma\Pi)^{(k+1)}$ formula of size and bit-complexity at most $\mathrm{poly}(s, D, t)$ and from Theorem 2.3.13, this can be done in at most $(sDt)^{O((sDt)^{\epsilon})}$ time for the arbitrary constant ϵ chosen in the beginning. Thus, the total time taken is at most $(sDt)^{O(d(sDt)^{\epsilon d})}$. □

Lemma 2.5.2 and Lemma 2.5.3 together imply our main theorems Theorem 1.7.1 and Theorem 1.7.2.

2.6 DEFERRED PROOFS

Circuit/formula bit-complexity

Proof of Lemma 2.3.2. We will prove an equivalent statement: the numerator and denominator of $f(\overline{a})$ have absolute value at most $2^{s \cdot b}$. We prove this by induction on the size of the formula. We will use $N(\cdot)$ and $D(\cdot)$ to denote the numerator and denominator of some rational number.

Base case: when $\mathrm{size}(C) = 1 = s$, there is a single leaf node in the formula that reads and outputs a single rational number of bit-complexity b, thus $\mathrm{bit}(f(\overline{a})) = b \leqslant s \cdot b$. The induction hypothesis is that for all formulas C with $\mathrm{size}(C) \leqslant S$ (for some $S \geqslant 1$), $\mathrm{bit}(f(\overline{a})) \leqslant \mathrm{bit}(C) \cdot b$. For the induction step, we look at formulas C with $\mathrm{size}(C) = S + 1$, and we consider two cases:

1. When the top gate is a sum gate: $f = \sum_{i=1}^{k} \alpha_i g_i(\overline{x})$, with C_i being the formula computing g_i and α_is being scalars from $\mathbb{Q}$.

$$
\begin{aligned}
f(\overline{a}) &= \sum_{i=1}^{k} \alpha_i g_i(\overline{a}) \\
|D(f(\overline{a}))| &= \left| \prod_{i=1}^{k} D(\alpha_i) D(g_i(\overline{a})) \right| \\
&\leqslant \prod_{i=1}^{k} 2^{\mathrm{bit}(\alpha_i)} 2^{\mathrm{bit}(g_i(\overline{a}))} \\
&\leqslant \prod_{i=1}^{k} 2^{\mathrm{bit}(\alpha_i)} 2^{\mathrm{bit}(C_i) \cdot b} && \text{(induction hypothesis)} \\
&= 2^{\sum_{i=1}^{k} (\mathrm{bit}(\alpha_i) + \mathrm{bit}(C_i) \cdot b)} \leqslant 2^{\mathrm{bit}(C) \cdot b}
\end{aligned}
$$

$$
\begin{aligned}
|N(f(\overline{a}))| &\leqslant \sum_{i=1}^{k} |N(\alpha_i)|\,|N(g_i(\overline{a}))| \prod_{j\neq i} \left|D(\alpha_j)\right| \left|D(g_j(\overline{a}))\right| \\
&\leqslant \sum_{i=1}^{k} 2^{\mathrm{bit}(\alpha_i)+\mathrm{bit}(g_i(\overline{a}))} 2^{\sum_{j\neq i}\mathrm{bit}(\alpha_j)+\mathrm{bit}(g_j(\overline{a}))} \\
&\leqslant \sum_{i=1}^{k} 2^{\mathrm{bit}(\alpha_i)+\mathrm{bit}(C_i)\cdot b} 2^{\sum_{j\neq i}\mathrm{bit}(\alpha_j)+\mathrm{bit}(C_j)\cdot b} \qquad \text{(induction hypothesis)} \\
&\leqslant \sum_{i=1}^{k} 2^{\sum_{j=1}^{k}\mathrm{bit}(\alpha_j)+\mathrm{bit}(C_j)\cdot b} \leqslant 2^{k+\sum_{j=1}^{k}\mathrm{bit}(\alpha_j)+\mathrm{bit}(C_j)\cdot b} \leqslant 2^{\mathrm{bit}(C)\cdot b}
\end{aligned}
$$

Thus, $\mathrm{bit}(f(\overline{a})) = \max\{\mathrm{bit}(N(f(\overline{a})), D(f(\overline{a})))\} \leqslant \mathrm{bit}(C)\cdot b$.

2. When the top gate is a product gate: $f = \prod_{i=1}^{k} \alpha_i g_i(\overline{x})$. The proof for the denominator in the case of sum gate will work here for both the numerator and the denominator. The required bound follows.

□

Relevant subclasses of algebraic circuits

Proof of Observation 2.3.5. We prove the size upper bounds here; the bit-complexity upper bounds proceed along exactly the same lines. The size upper bound for the sum is immediate and hence we only need to focus on the product. Let the expressions for P and Q be

$$
\begin{aligned}
P_r &= \sum_i P_{r,i}^{(r)} \cdot g_{r,i}^{a_{r,i}} \\
\implies \prod P_r &= \sum_{r_1,\ldots,r_t} (P_{1,r_1}\cdots P_{t,r_t}) \cdot \left(g_{1,r_1}^{a_{1,r_1}} \cdots g_{t,r_t}^{a_{t,r_t}}\right)
\end{aligned}
$$

where each $P_{i,j}$ is computed by $(\Sigma\Pi)^k$ formulas of size at most s, and each $g_{i,j}$ is a polynomial of degree at most d.

Each $(P_{1,r_1}\cdots P_{t,r_t})$ is computed by a $(\Sigma\Pi)^k$ formula of size at most s^t. By Lemma 2.3.9, $g_{1,r_1}^{a_{1,r_1}} \cdots g_{t,r_t}^{a_{t,r_t}}$ can be expressed as a sum $\sum_{\ell=1}^{s^t} f_\ell^D$ where $D = \sum_j a_{j,r_j}$ and each f_ℓ is a degree polynomial of degree at most d. Thus, $\prod_r P_r$ is computable by a $\Sigma\left((\Sigma\Pi)^{(k)} \cdot (\mathrm{Deg}_d)^*\right)$ formula of size at most $s^{O(t)}$. □

Polynomial identity testing

Proof of Lemma 2.3.12. Since f is a non-zero polynomial of degree at most d, there is a monomial $\overline{x}^{\overline{e}}$ of degree at most d with a non-zero coefficient in f. Let S be the support of the monomial $\overline{x}^{\overline{e}}$, i.e., $S = \{x_i : e_i \neq 0\}$. Clearly, $|S| \leqslant d$. We now consider the polynomial $\tilde{f}$ obtained from f by setting all the variables x_j not in the set S to zero. Since f has a non-zero monomial with support contained in the set S, $\tilde{f}$

continues to be a non-zero polynomial of degree at most d. Moreover, it is a d variate polynomial since it only depends on the variables in S. From Lemma 2.3.10, we get that for any subset T_d of $\mathbb{Q}$ of cardinality at least $d+1$, there exists a vector $\overline{b} \in {T_d}^d$ such that $\tilde{f}(\overline{b}) \neq 0$. Let $\overline{a} \in \mathbb{Q}^n$ to be such that for every $i \in S$, $a_i = b_i$ and for every $i \notin S$, $a_i = 0$. Then, $f(\overline{a}) = \tilde{f}(\overline{b}) \neq 0$. Moreover, $\overline{a}$ is in $\mathcal{H}(d, n)$. □

Proof of Lemma 2.3.15. Let T_d be the set $\{0, 1, 2, 3, \ldots, d\}$ and let $\mathcal{H}(d, n)$ be the set of points defined in Definition 2.3.11, i.e.,

$$\mathcal{H}(d,n) = \left\{ (a_1, \ldots, a_n) \ : \ S \in \binom{[n]}{\leqslant d}, \ a_i \in T_d \text{ for all } i \in S \text{ and } a_j = 0 \text{ for all } j \notin S \right\}.$$

From Lemma 2.3.12, we know that every non-zero polynomial f of degree at most d must evaluate to non-zero on some point of $\mathcal{H}(d, n)$. In other words, two distinct degree d polynomials f and g cannot agree on every point of $\mathcal{H}(d, n)$. An immediate consequence of this is that if we are given the evaluations of an unknown polynomial f on all points of $\mathcal{H}(d, n)$, and we view each of these evaluations as a linear constraint on the unknown coefficients of f, then this linear system has a unique solution.

Based on this observation, a natural algorithm for computing the coefficient vector of C is the following, we evaluate the given formula on every input in $\mathcal{H}(d, n)$, set up the linear system on the coefficients of C obtained from these evaluations, and use any standard linear system solver over $\mathbb{Q}$ to solve this system.

Note that the size of this linear system is at most $n^{O(d)}$, and from Lemma 2.3.2, of the constants in this linear system is at most $\text{poly}(s, b, d)$. Thus, this linear system can be solved in time $\text{poly}(s, b, d, n^d) \leqslant \text{poly}(s, b, n^d)$ time as claimed. □

Deterministic divisibility testing and PIT

Proof of Corollary 2.3.19. The proof essentially follows immediately from Theorem 2.3.18. From Theorem 2.3.18, we have that g divides f if and only if $R(\overline{x}) := f(\overline{x}) - g(\overline{x})Q(\overline{x}) \equiv 0$, where Q is the pseudo-quotient of f and g. It suffices to show that $R(\overline{x})$ has $\mathcal{C} = \Sigma\left((\Sigma\Pi)^k \cdot (\text{Deg}_d)^*\right)$ formulas of size $\text{poly}(s, d)$, and since $f(\overline{x}) \in (\Sigma\Pi)^k$, it suffices to bound the size of $\mathcal{C}$-formulas computing $g(\overline{x}) \cdot Q(\overline{x})$.

By Lemma 2.3.17, the pseudo-quotient $Q(\overline{x})$ is computable by $\mathcal{C}$-formulas of size $\text{poly}(s, d)$. Let one such computation be of the form

$$Q(\overline{x}) = \sum_i f_i \cdot g_i^{e_i} \qquad \text{where each } f_i \in (\Sigma\Pi)^k \text{ and } \deg(g_i) \leqslant d$$

$$\text{and each } e_i \leqslant s$$

$$\implies g(\overline{x})Q(\overline{x}) = \sum_i f_i \cdot (g \cdot g_i^{e_i})$$

From Lemma 2.3.9, note that any term of the form $(g \cdot h^e)$ can be expressed as

$$g \cdot h^e = \sum_{i=1}^{\mathrm{poly}(e)} \beta_i \cdot (g + \alpha_i h)^{e+1}$$

for field constants α_i's and β_i's. Thus, feeding this in the above expression for $g \cdot Q$, we have

$$g(\overline{x}) \cdot Q(\overline{x}) = \sum_i \sum_j f_i \cdot \tilde{g}_{ij}^{e_{ij}}$$

for polynomial $\tilde{g}_{ij}$ of degree at most d, and thus is also a $\mathcal{C}$-formula of size at most $\mathrm{poly}(s, d)$. Therefore, $R(\overline{x}) = f(\overline{x}) - g(\overline{x})Q(\overline{x})$ is also computable by $\mathcal{C}$-formulas of size $s' = \mathrm{poly}(s, d)$. Thus, we can check if g divides f by checking if $R(\overline{x}) \equiv 0$ (by Theorem 2.3.18) which can be done in $T(k, d, s')$ time as claimed. □

Hensel lifting

Proof sketch of Lemma 2.3.26. As indicated earlier, the lemma is almost an immediate consequence of Lemma 3.6 in [KSS15]. The precise statement there gives a circuit $\tilde{C_k}$ of size and bit-complexity $\mathrm{poly}(s, D, 2^k)$ for g_k, h_k. We notice that without loss of generality, the degree of g_k, h_k and hence of $\tilde{C_k}$ can be assumed to be at most $(D + 2^k)$ since the y degree is at most D and the x degree is at most 2^k. This incurs at most a polynomial blow up in the circuit size.

Now, to go from circuits for g_k, h_k to formulas computing these polynomials, we just invoke the classic depth reduction result of Valiant, Skyum, Berkowitz and Rackoff [Val+83] to get a formula of size (and bit-complexity) at most $\mathrm{poly}(s, D, 2^k)^{\log(D+2^k)} \leqslant (sDk)^{k \log D}$.

Note that if we had a better bound of d on the total degree of g_k, then the size and bit-complexity of the formula for g_k is at most $(sDk)^{O(\log d)}$. □

3 LIST-FACTORING

3.1 OVERVIEW

In the previous chapter, we discussed an algorithm that outputs all low-degree factors of a constant-depth circuit. In an attempt to improve this algorithm so that it outputs *all* factors of a constant-depth circuit, we obtain the following stepping-stone result, which is the main result of this chapter.

Theorem 1.7.5. *[Subexponential list containing constant-depth factors of constant-depth circuits] Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $\Delta_1, \Delta_2 \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta_1)}$ of size s, bit-complexity t and degree D, along with a size parameter m, and outputs a list of circuits $L = \{C_1, \ldots, C_r\}$ with the following properties.

1. *Each $C_i \in L$ is an arithmetic circuit of size $\mathrm{poly}(s, m, D)$ and bit-complexity $\mathrm{poly}(t, s, D, m)$, with division gates.*
2. *For every irreducible factor g of C with a size m $(\Sigma\Pi)^{(\Delta_2)}$-circuit computing it, there exists a $C_i \in L$ such that C_i computes g.*

The size of the list L as well as the running time of the algorithm are bounded above by $O(smDt)^{O((smDt)^{\epsilon})}$.

We remark that indeed Theorem 1.7.5 doesn't completely solve the problem of finding sparse (or more generally, low depth) factors of low depth circuits. It merely outputs a "short" (subexponential) list L of candidate polynomials such that every irreducible factor of the input is contained in L. As explained in Section 3.2, for technical reasons, we currently do not know how to deterministically prune the list L to obtain the true factors.

Yet another technicality is that our algorithm needs a size bound on the size of a circuit for the factors we care about because no structural guarantees are known for the factors of constant-depth circuits. For instance, we do not know if the factors of a constant-depth circuit also have small constant-depth circuits.

A SLIGHTLY MORE GENERAL STATEMENT. Our techniques in the proof of Theorem 1.7.5 give a slightly more general statement connecting the questions of deterministic polynomial factorisation and deterministic polynomial identity testing. Let $\mathcal{C}$ be a class of circuits

that are somewhat rich in the sense that if a circuit C is in $\mathcal{C}$, then the circuit C' obtained from C by a change of basis continues to be in $\mathcal{C}$, and for circuits $C_1, C_2, \ldots, C_n$ in $\mathcal{C}$, and a circuit $B(y_1, y_2, \ldots, y_n)$ of constant depth, the circuit $B(C_1, C_2, \ldots, C_n)$ continues to be in $\mathcal{C}$. For instance, the class of polynomials computable by small constant-depth circuits, or by small formulas satisfy the above property. Our techniques in the proof of Theorem 1.7.5 generalize to such classes and give a deterministic reduction from the question of computing irreducible factors of a polynomial $f \in \mathcal{C}$ that are also computable in $\mathcal{C}$ to the question of polynomial identity testing of polynomials (of slightly larger complexity) in $\mathcal{C}$; the major caveat being that the algorithm will only output a list containing all such irreducible factors of interest, but might also contain some spurious circuits. Moreover, these circuits in the output list are potentially of unbounded depth, can contain division gates and some of these spurious circuits in the list might not even be well defined, i.e., they involve division by an identically zero polynomial. In contrast to this, the prior known connections between PIT and polynomial factorisation, most notably in the work of Kopparty, Saraf and Shpilka [KSS15] gives a truly deterministic polynomial factorisation algorithm, given access to an oracle for PIT. However, even if the input to the polynomial factorisation algorithm is very structured (for instance, is a sparse polynomial), the PIT instances generated in the process appear to be very general.

In spite of this slight generality, for a clean and complete presentation of the results, we only work with constant-depth circuits in the rest of the chapter.

FIELD DEPENDENCE OF OUR RESULT. For our results, we work over the field of rational numbers due to various technical reasons. Our proofs rely on derivative-based techniques like working with the Taylor expansion of a polynomial, and working over fields of characteristic zero ensures that derivatives do not inadvertently vanish. We also need a deterministic algorithm for univariate factorisation over the underlying field as a subroutine for our algorithms. Moreover, we also need a non-trivial deterministic algorithm for polynomial identity testing of constant-depth circuits over the underlying field [LST25]. The field of rational numbers satisfies all these requirements.

SUBSEQUENT WORK. A few weeks after a preliminary version of this work appeared on arxiv, Andrews and Wigderson announced a beautiful and surprising result [AW24], in which they show that many natural linear algebraic problems, such as computing the resultant of two polynomials, their gcd, their lcm, etc., can be done using constant-depth arithmetic circuits. Their result is quite general and provides a way of obtaining a strengthening of results; our proof is more specific to the problem of derandomizing factorisation. Using [AW24], we

obtain a subexponential time deterministic algorithm that takes a constant-depth circuit as input and output a *polynomial* size list of polynomial sized circuits with division gates, such that every factor (and not just every constant-depth factor) of the input polynomial has a representative in the output list, along with potential spurious factors.

Theorem 1.7.6. *[Polynomial-sized list containing constant-depth factors of constant-depth circuits, in subexponential time] Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $\Delta \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta)}$ of size s, bit-complexity t and degree D, and outputs a list of circuits $L = \{C_1, \ldots, C_r\}$ with the following properties.

1. *Each $C_i \in L$ is an arithmetic circuit of size $\mathrm{poly}(s, D)$ and bit-complexity $\mathrm{poly}(t, s, D)$, with division gates.*

2. *For every irreducible factor g of C, there exists a $C_i \in L$ such that C_i computes g.*

The size of the list L is $\mathrm{poly}(d)$. The running time of the algorithm is bounded above by $O(sDt)^{O((sDt)^{\epsilon})}$.

In the rest of the chapter, we mostly focus on the proof of Theorem 1.7.5; the proof of Theorem 1.7.6 follows mostly along similar lines with some minor modifications, and we provide a sketch of the algorithm in the next chapter.

3.2 PROOF OVERVIEW

In this section, we give a brief overview of the main ideas in our proof. But first, we set up some notation: f denotes the input polynomial of degree d that is given via a small algebraic circuit of size s and depth Δ computing it. Furthermore, we assume that f factors as $f = g^m \cdot h$, where g is an irreducible polynomial and can be computed by a small constant-depth circuit, and g and h do not have a non-trivial GCD. The assumption is without loss of generality: the case when f is a perfect power of an irreducible polynomial g is implicitly handled since we work with all the derivatives of f; moreover, there are other direct ways known in the literature to handle this case (see [ST20]). Throughout the chapter, the underlying field is assumed to be the field $\mathbb{Q}$ of rational numbers. We note that a priori, it is not clear if h can also be computed by a small constant-depth circuit. However, this follows immediately from an observation in a recent prior work of Kumar, Saptharishi and Ramanathan [KRS23] (see Lemma 2.3.17 in Chapter 2).

We now outline the sketch of our ideas. Our algorithm follows the general outline of a typical factorisation algorithms in the literature

(e.g., [Kal89]). However, in their original formulations, some of these steps rely on randomness, and we elaborate how we get around this and implement the step in a deterministic way.

Making f monic: As the first step of the algorithm, we make the polynomial f monic in one of the variables, which we denote by y here by applying an invertible linear transformation to the variables. This is typically done by choosing field constants $\overline{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{F}^n$ at random, and replacing the variable x_i in the circuit for f by the linear form $x_i + a_i y$. Since the underlying field is large enough, via the Schwartz-Zippel lemma, we get that the coefficient of y^d in $f(\overline{x} + \overline{a} \cdot y)$ is a non-zero field constant with high probability, and up to a scaling by this field constant, we obtain a polynomial that is monic in y. For the ease of notation, we continue to denote this new polynomial by f. To derandomize this step, we note that the coefficient of y^d in $f(\overline{x} + \overline{a} \cdot y)$ is just the evaluation of the homogeneous component of $f(\overline{x})$ of degree d on the input $\overline{a}$. But since f has a size s, depth Δ circuit, its degree d homogeneous component has a circuit of size $s' = \text{poly}(s, d)$ and depth $\Delta' = \Delta + O(1)$ (see Lemma 2.3.7). Thus, if we try all possible inputs $\overline{a}$ from a hitting set for size s', depth Δ' circuits, one of the vectors will have the desired property. A deterministic subexponential time construction of such a hitting set for constant-depth circuits was shown by Limaye, Srinivasan and Tavenas [LST25].

For the rest of the algorithm, we view f as a polynomial in $\mathbb{Q}[\overline{x}][y]$. We also note that since f is monic in y, without loss of generality, g and h can be assumed to be monic in y (Lemma 3.3.4).

Reducing the multiplicity of g in f: The remaining steps of the algorithm assume that the multiplicity of g in f, denoted by the parameter m here, is one. This can be assumed without loss of generality. Indeed, to reduce to this case, we run our algorithm on all the partial derivatives of f with respect to y, i.e. the polynomials $\{\frac{\partial^i f}{\partial y^i} : i \in \{0, 1, \ldots, d-1\}\}$; the observation being that g is an irreducible factor of multiplicity one of at least one of these polynomials. Moreover, all these polynomials can be computed by a circuit of size $\text{poly}(s, d)$ and depth Δ (Lemma 2.3.8). So, it suffices to focus on recovering factors of multiplicity one of f. This observation was also used in [KRS23].

Finding a good starting point for Newton Iteration: The goal now is to view f as a univariate in y with coefficients from $\mathbb{Q}[\overline{x}]$ and obtain a root $\Phi(\overline{x})$ of f in the ring $\mathbb{Q}[[\overline{x}]]$ of power series in $\overline{x}$. Moreover, in order to recover the factor g of f using this root, it must be the case that this is a root of g (and hence a root of f). In fact, by standard techniques, it suffices to recover an approximation of Φ with high enough accuracy, i.e., to recover $\Phi_k(\overline{x}) := \Phi(\overline{x}) \mod \langle \overline{x} \rangle^k$ for some $k > 2d^2$. This is

done via Newton iteration, where we start from the constant term (i.e. $\Phi_0 := \Phi(\overline{x}) \mod \langle \overline{x} \rangle$) of such a Φ, and *lift* it to obtain Φ_k. While the lifting process is itself deterministic (see Lemma 3.3.1), a crucial issue with respect to the starting point of the lifting process is the following: Φ_0 must be a root of $f(\overline{x}, y) \mod \langle \overline{x} \rangle$ of multiplicity one, and in fact must come from $g \mod \langle \overline{x} \rangle$. This is typically done by translating the $\overline{x}$ variables to $\overline{x} + \overline{a}$ such that the univariate polynomials $g(\overline{a}, y)$ and $h(\overline{a}, y)$ do not share a non-trivial GCD and $g(\overline{a}, y)$ is square-free. This ensures that every root of $f(\overline{a}, y) = g(\overline{a}, y)h(\overline{a}, y)$ that comes from $g(\overline{a}, y)$ is a root of multiplicity one of $f(\overline{a}, y)$ and hence is a good starting point for lifting via Newton Iteration.

In typical factorisation algorithms, such a point $\overline{a}$ is found by first preprocessing the input f such that it is square-free, and then setting $\overline{a}$ to be a non-zero of the discriminant of f. Since the discriminant is an efficiently computable low-degree polynomial (assuming f is), by the Schwartz-Zippel lemma $\overline{a}$ can be chosen at random. This guarantees that $f(\overline{a}, y) = g(\overline{a}, y)h(\overline{a}, y)$ is square-free and hence $g(\overline{a}, y)$ and $h(\overline{a}, y)$ don't have a non-trivial GCD. In our setting, we do not have the guarantee that $f(\overline{x}, y)$ is square-free since h might not be square-free, but we know that g is a multiplicity one irreducible factor of f. Moreover, we would like to find such a point $\overline{a}$ deterministically. A serious quantitative issue with respect to derandomizing the application of the Schwartz-Zippel lemma here is that even though f has a small constant-depth circuit, its discriminant is the determinant of a matrix of dimension roughly $2d$ with polynomial entries, a powerful model for which no non-trivial deterministic PIT algorithms are known. This is also the issue if we try to look at the resultant of g and h instead.

One intriguing possibility here would be to show that the discriminant or the resultant of polynomials with small constant-depth circuits have relatively small constant-depth circuits. However, we do not know how to show this[1]. Instead, we come up with an alternative polynomial $R_{f,g}$, referred to as the *pseudo-resultant* of f and g, and rely on its properties to arrive at our good starting point for the lifting process. In particular, we note that the complexity of $R_{f,g}$ is close to the complexity of f and g. We then show that when g is an irreducible factor of f with multiplicity one, $R_{f,g}(\overline{x}, y) \not\equiv 0$. Moreover, using ideas from a divisibility testing algorithm of Forbes [For15], we can show that for any $\overline{a} \in \mathbb{Q}^n$ such that $R_{f,g}(\overline{a}, y) \not\equiv 0$, there is a root of multiplicity one of $f(\overline{a}, y)$ that is a root of $g(\overline{a}, y)$ but not $h(\overline{a}, y)$. This root becomes a good starting point for the lifting process. Furthermore, since $R_{f,g}$ is computable by a small constant-depth circuit, such a point $\overline{a}$ can be obtained deterministically in subexponential time using the results in [LST25].

1 Now, Andrews and Wigderson [AW24] have shown that the resultant and the discriminant of polynomials with small constant-depth circuits indeed do have small constant-depth circuits; see the remark in Section 3.2.1.

Intuitively, while the randomized algorithm for finding a good starting point for lifting uses the resultant to find an $\overline{a}$ such that $g(\overline{a}, y)$ and $h(\overline{a}, y)$ do not have a non-trivial GCD, we observe that this is a bit of an overkill, and it suffices to find an $\overline{a}$ such that the GCD of $g(\overline{a}, y)$ and $h(\overline{a}, y)$ is not equal to $g(\overline{a}, y)$. Equivalently, it suffices to find an $\overline{a}$ such that $g(\overline{a}, y)$ does not divide $h(\overline{a}, y)$. This happens to be a computationally easier condition to achieve deterministically via the pseudo-resultant. This is essentially the main technical idea in this work, and is formally discussed in Section 3.4.

One undesirable quantitative issue here is that the circuits obtained for Φ via Newton iteration, while being of polynomially bounded size, can have unbounded depth. This is one of the reasons that the output list of our algorithm can have spurious circuits not corresponding to factors of f.

Solving a linear system to obtain g: Once we have obtained an approximate root $\Phi_k(\overline{x})$ of $f(\overline{x}, y)$, we would like to recover the real factor $g(\overline{x}, y)$. If the y-degree of g is known to be d', then this is done via thinking of g as $g(\overline{x}, y) = y^{d'} + \sum_{i=0}^{d'-1} g_i(\overline{x}) y^i$, where g_i is a formal variable, and imposing the constraint that

$$g(\overline{x}, \Phi_k(\overline{x})) = \phi_k(\overline{x})^{d'} + \sum_{i=0}^{d'-1} g_i(\overline{x}) \Phi_k(\overline{x})^i \equiv 0 \mod \langle \overline{x} \rangle^k .$$

Furthermore, each g_i can be written as $g_i(\overline{x}) = \sum_{j=0}^{d} g_{i,j}(\overline{x})$, where $g_{i,j}$ is the homogeneous component of g_i of degree j. The constraint above translates naturally into a linear system in the variables $g_{i,j}$, and the factor g does satisfy this system of equations. Furthermore, it can be shown that if d' equals the y-degree of g and $k > 2d^2$, then g is the unique solution to this system of linear equations. Moreover, if we set a similar system with the y-degree of the factor of interest being set to something less than d', then the resulting linear system has no solution. We rely on these properties of the linear system to obtain an algebraic circuit for g. Essentially, the idea is that if we know d' and the resulting linear system is given by a square matrix, then uniqueness of solution of degree d' guarantees that this matrix is invertible, and the solution can then be expressed as an algebraic circuit given by Cramer's rule. However, the constraint matrix of the system might not be square in general, and in this case, we first make the system square (deterministically) before applying Cramer's rule. The details can be found in Lemma 3.3.3.

One added subtlety here is that in general, we would not know the exact y-degree of g. In this case, we try all values from 1 to $d-1$, and for each one formally construct an algebraic circuit with divisions and unbounded depth (as noted earlier, the circuit for Φ can already have

unbounded depth) and include all such circuits in our final list of solutions.

DIFFERENCES WITH [KRS23]: Many of the high-level ideas in the algorithm outlined above are similar to those in a prior work of Kumar, Ramanathan and Saptharishi [KRS23], where deterministic subexponential time algorithms were given for computing constant-degree factors of constant-depth circuits. The similarities include the motivation for the problems as well as the high-level approach of understanding the structure of the PIT instances appearing in the randomized algorithms for multivariate factorisation for these special cases carefully and derandomizing those steps. However, there are key technical differences. The main technical result of [KRS23] is an upper bound on the complexity of the resultant of a constant-depth circuit and a constant-degree polynomial. However, we are unable to prove such an upper bound in our case. Instead, we define the pseudo-resultant and observe that its circuit complexity is comparable to that of the given input polynomial and the factors we care about. Moreover, it helps us completely avoid working with the resultant of g and h or the discriminant of f, and lets us find a good starting point for Newton iteration in deterministic subexponential time. There are also some technical differences in how we go from approximate power series roots to actual factors of f, but the main takeaway from this work is perhaps the notion of pseudo-resultant and its properties.

INABILITY TO PRUNE THE LIST: Unlike the results in [KRS23], where only true low-degree factors of the input polynomial are output, we end up with a list of circuits containing the circuits for all irreducible factors with small constant-depth circuits, but perhaps much more. This difference stems from the fact that in the algorithm of [KRS23], at an intermediate step a list of constant-degree polynomials (essentially expressed as a sum of monomials) is constructed, where similar to this work, the list contains all the true constant-degree factors, as well as some spurious factors. However, since in [KRS23] every polynomial in this intermediate list is a constant-degree polynomial in the monomial basis, the list is pruned to true factors by checking if a polynomial in the list divides the given input polynomial, and whether they are irreducible (which can be done deterministically since the degree of the polynomial is a constant). However, in this work, the list consists of unbounded depth algebraic circuits with division gates. For such circuits, we do not even know how to test if the circuit is well-defined, i.e., that there are no denominators that are identically zero. Hence, it is unclear to us if the list can be pruned to the true irreducible factors in a non-trivial way deterministically. Addressing this issue seems like a very interesting open problem.

3.2.1 Related work and subsequent progress

RECENT RESULTS OF ANDREWS AND WIGDERSON [AW24], AND SOME CONSEQUENCES: As mentioned in the introduction, Andrews and Wigderson [AW24] show that many natural linear algebraic problems, such as computing the resultant of two polynomials, their gcd, their lcm, etc., can be done using constant-depth arithmetic circuits. A key insight of their work is a result that shows that one can implement the transformation between elementary symmetric polynomials and power sum polynomials efficiently via constant-depth circuits; moreover, they use these efficient circuits to manipulate the multiplicities of various roots of a polynomial while only having access to the coefficients of the polynomial. We define the pseudo-resultant as a way to bypass the need to bound the complexity of the resultant of polynomials computed by constant-depth circuits. But, Andrews and Wigderson show that the resultant and discriminant are themselves computable using constant-depth circuits. As a consequence of the same ideas, they also show that the squarefree decomposition of a constant-depth circuit can be computed in subexponential time, with the guarantee that each part of the decomposition (corresponding to factors of multiplicity $i \in [D]$) will have constant-depth circuits. Instead of using a hitting set for the pseudo-resultant, we can first compute the squarefree decomposition and use a hitting-set generator on the discriminant of each polynomial in the squarefree decomposition. In particular, we exactly know the polynomials that we want to hit, as opposed to our current strategy, where we iterate over all the points in the hitting set so that one of them works for any particular factor. With this change, our algorithm can be guaranteed to output a polynomially large list that contains *all* factors, as opposed to a subexponentially large list of candidate factors that contains all constant-depth factors. We give a high-level sketch of this algorithm in the next chapter.

3.3 PRELIMINARIES

We shall briefly discuss some preliminaries. A reader familiar with the literature can skip to Section 3.4 and come back if needed.

CIRCUITS OVER NUMBER FIELDS: In our proofs, we encounter circuits over fields of the form $\mathbb{Q}(\alpha)$ where α is some algebraic number. For such circuits, we think of their description of each field constant as an element of $\mathbb{K} = \frac{\mathbb{Q}[u]}{A(u)}$ where $A(u)$ is its minimal polynomial of α. The bit-complexity of any element of $\mathbb{Q}(\alpha)$ is defined as the bit-complexity of the unique representation as a element of $\mathbb{K}$. The bit-complexity of such a circuit is the total description size of the circuit, which includes the bit-complexity of any constants used internally.

3.3.1 Newton iteration and approximate roots

The following are some standard facts about Newton iteration and approximate roots.

Lemma 3.3.1 (Newton iteration)**.** *Let* $F(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ *be monic in* y*. Suppose* $u \in \mathbb{F}$ *such that*

$$\begin{aligned} F(\mathbf{o}, u) &= 0, \\ (\partial_y F)(\mathbf{o}, u) &\neq 0. \end{aligned}$$

Then, for all $k \geqslant 0$*, there is an* approximate root *polynomial* $\Phi_k \in \mathbb{F}[\overline{x}]$ *that satisfies*

$$\begin{aligned} F(\overline{x}, \Phi_k) &= 0 \bmod \langle \overline{x} \rangle^{k+1}, \\ \Phi_k(\mathbf{o}) &= u. \end{aligned}$$

Furthermore, if we are provided F *a circuit of size and bit-complexity* s*, and also the element* $u \in \mathbb{F}$ *with bit-complexity at most* B*, we can output an algebraic circuit for the approximate root* $\Phi_k(\overline{x})$ *of size at most* $\mathrm{poly}(s, k)$ *and bit-complexity* $\mathrm{poly}(s, B, k)$*.*

Proof. The approximate roots are given by the following recursive definition:

$$\begin{aligned} \Phi_0 &= u \\ \text{For all } k \geqslant 0, \quad \Phi_{k+1} &= \Phi_k - \frac{F(\overline{x}, \Phi_k)}{\partial_y F(\mathbf{o}, u)}. \end{aligned}$$

It is clear from the above definition that Φ_{k+1} is a polynomial in $\overline{x}$ and its circuit complexity is at most an additive $O(s)$ larger than the circuit size of Φ_k. Since the only constant introduced in the circuit is $(\partial_y F(\mathbf{o}, u))^{-1}$, the bit-complexity is bounded by an additive $\mathrm{poly}(s, B)$.

We now show that Φ_{k+1} satisfies the requirement by induction (the base case of Φ_0 is trivial). By Taylor expansion around $(\overline{x}, \Phi_k)$, if $z \in \langle \overline{x} \rangle^{k+1}$, we have

$$\begin{aligned} F(\overline{x}, \Phi_k + z) &= F(\overline{x}, \Phi_k) + z \cdot \partial_y F(\overline{x}, \Phi_k) \\ &\quad + \left(\frac{z^2}{2!}\right) \partial_{y^2} F(\overline{x}, \Phi_k) + \cdots \\ &= F(x, \Phi_k) + z \cdot \partial_y F(\overline{x}, \Phi_k) \bmod \langle \overline{x} \rangle^{k+2} \quad (\text{since } z^2 \in \langle \overline{x} \rangle^{2(k+1)} \subseteq \langle \overline{x} \rangle^{k+2}). \end{aligned}$$

Furthermore, since $z \in \langle \overline{x} \rangle^{k+1}$, we have that

$$z \cdot \partial_y F(\overline{x}, \Phi_k) = z \cdot \partial_y F(\mathbf{o}, \Phi_k(0)) \bmod \langle \overline{x} \rangle^{k+2}$$

since the other terms from the second multiplicand only contribute higher degree terms in $\overline{x}$. Thus,

$$\begin{aligned} F(\overline{x}, \Phi_k + z) &= F(x, \Phi_k) + z \cdot \partial_y F(\mathbf{o}, u) \bmod \langle \overline{x} \rangle^{k+2}, \\ &= 0 \bmod \langle \overline{x} \rangle^{k+2} \quad \text{when } z = -\frac{F(x, \Phi_k)}{\partial_y F(\mathbf{o}, u)}, \\ \implies F(\overline{x}, \Phi_{k+1}) &= 0 \bmod \langle \overline{x} \rangle^{k+2}. \end{aligned}$$

□

The following lemmas are standard in the literature (see [Büro4]) and we include proofs for completeness.

Lemma 3.3.2 (Minimal polynomials of approximate roots)**.** *Let $\mathbb{F}$ be a field of characteristic zero. Let $\phi(t) \in \mathbb{F}[t]$ and $G(t, y) \in \mathbb{F}[t, y]$ be polynomials such that G is irreducible of y-degree d_y, t-degree d, is monic in y and satisfies*

$$G(t, \phi(t)) \equiv 0 \mod \langle t \rangle^k,$$

for some natural number k greater than $2d_y d$. If $H(t, y) \in \mathbb{F}[t, y]$ is a non-zero polynomial of y-degree exactly d_y, t-degree d, is monic in y and satisfies

$$H(t, \phi(t)) \equiv 0 \mod \langle t \rangle^k,$$

then, H must be equal to G.

Proof. Let $R(t) = \mathrm{Res}_y(G(t, y), H(t, y)) \in \mathbb{F}[t]$. From the definition of the resultant, the degree of R is at most $2d_y d$. If R is identically zero, then we have from Lemma 2.3.22 that G and H have a non-trivial GCD as polynomials in $\mathbb{F}(t, y)$. But then, since G and H are monic in y, it follows from Lemma 3.3.4 that they must have a non-trivial GCD in $\mathbb{F}[t, y]$. Now, since G is irreducible, this can happen only if G divides H. Since the degree of H is at most d, we get that they must be a non-zero constant multiple of each other. But they are both monic in y and hence they must be equal to each other. Thus, if R is identically zero, then we are done. It now remains to show that R is indeed identically zero.

We have from Lemma 2.3.22 that there exist polynomials $A(t, y), B(t, y)$ such that

$$A(t, y)G(t, y) + B(t, y)H(t, y) = R(t).$$

Now, plugging in $y = \phi(t)$ on both sides of the above equation, we get

$$A(t, \phi(t))G(t, \phi(t)) + B(t, \phi(t))H(t, \phi(t)) = R(t).$$

But from the hypothesis of the lemma, we know that $G(t, \phi(t)) \equiv 0 \mod \langle t \rangle^k$ and $H(t, \phi(t)) \equiv 0 \mod \langle t \rangle^k$. Moreover, $k > 2d_y d$. Thus, the left-hand side of the identity above vanishes modulo $\langle t \rangle^{2d_y d+1}$. So, the resultant R vanishes modulo $\langle t \rangle^{2d_y d+1}$. But since its degree is less than $2d_y d$, this means that R is identically zero as a polynomial in $\mathbb{F}[t]$. This completes the proof of the lemma. □

Lemma 3.3.3 (Computing minimal polynomials of approximate roots)**.** *Let* $\Phi_k(\overline{x}) \in \mathbb{F}[\overline{x}]$ *be provided by an algebraic circuit of size and bit-complexity at most* s*. Suppose there exists a polynomial* $G(\overline{x}, y) = \sum_{i=0}^{d} G_i(\overline{x}) y^i$ *that is monic in* y *and irreducible satisfying*

$$\begin{aligned} G(\overline{x}, \Phi_k(\overline{x})) &= 0 \bmod \langle \overline{x} \rangle^{k+1}, \\ \deg G &\leqslant D, \\ \deg_y G &= d, \\ k &\geqslant 2Dd. \end{aligned}$$

Then, given the exact y*-degree* d*, total degree* D*, and the circuit for* Φ_k*, for every* $i \in [d]$*, we can compute an algebraic circuit (with only* $\overline{x}$ *variables and no* y*), with divisions, of size and bit-complexity at most* $\mathrm{poly}(s, d, D)$ *for the polynomial* $G_i(\overline{x})$*. In particular, we can compute a circuit of size and bit-complexity* $\mathrm{poly}(s, d, D)$ *for* $G(\overline{x}, y)$*.*

Proof. Let us first do a standard transformation from the multivariate setting to the bivariate setting by mapping each x_i to $t \cdot x_i$, where t is a new variable. Thus, $G(\overline{x}, y)$ gets mapped to $\tilde{G}(t, y) \in \mathbb{F}[\overline{x}][t, y]$ and each monomial in G with $\overline{x}$-degree l gets mapped to a monomial with t-degree l in $\tilde{G}$. Similarly, $\Phi_k(\overline{x})$ gets mapped to $\tilde{\Phi}_k(t)$.

$$\tilde{G}(t, y) = \sum_{i=0}^{d} \left(\sum_{j=0}^{D} G_{ij} t^j \right) \cdot y^i$$

where each G_{ij} is the degree-j homogeneous part (in $\overline{x}$) of the coefficient of y^i in G. We will treat[2] each G_{ij} as an indeterminate from $\mathbb{F}(\overline{x})$ and express the given constraints as a system of linear equations in G_{ij}'s. Note that $\tilde{G}$, as a bivariate polynomial in (t, y), is irreducible because: 1) monicness along with Lemma 3.3.4 ensures that all factors are (without loss of generality) from $\mathbb{F}[\overline{x}][t, y]$, and 2) any such factor (which must also be monic in y), upon setting $t = 1$, will give us a non-trivial factor of $G(\overline{x}, y)$ in $\mathbb{F}[\overline{x}, y]$, which is a contradiction since G is irreducible.

2 This is an overload of notation: G_{ij}'s are the homogeneous parts of the polynomial G given in the lemma statement, as well as the indeterminates in the linear system that will eventually have the polynomial from the lemma statement as its unique solution.

For $r = 0, \ldots, k$, let $\tilde{\Phi}_k^r = \Phi_k^{(r,0)} + \Phi_k^{(r,1)} t + \cdots + \Phi_k^{(r,k)} t^r$ where $\Phi_k^{(r,\ell)}$ is the degree-ℓ homogeneous part of Φ_k^r. Given a circuit of size s for Φ_k, we can obtain poly(s)-sized circuits for each of $\Phi_k^{(r,\ell)}$.

$$\begin{aligned}\tilde{G}(t, \tilde{\Phi}_k) &= \sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \tilde{\Phi}_k^i t^j = \sum_{i=0}^{d} \sum_{j=0}^{D} \sum_{\ell=0}^{k} G_{ij} \Phi_k^{(i,\ell)} t^{\ell+j} \\ &= \sum_{m=0}^{D+k} \left(\sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \Phi_k^{(i,m-j)} \right) t^m\end{aligned}$$

where the last equality is just grouping the terms in terms of the t-degree. Thus, the given constraints can be expressed as

$$\begin{aligned}\left(\sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \Phi_k^{(i,m-j)} \right) &= 0 \quad \text{for all } m = 0, \ldots, k, \\ G_{d,0} &= 1, \\ G_{d,j} &= 0 \quad \text{for all } j > 0.\end{aligned}$$

By Lemma 3.3.2, the solution to the linear system is a unique polynomial in $\mathbb{F}(\overline{x})[t, y]$; thus, it follows that the column rank of the constraint matrix M of this linear system is full. Now, if the coefficient vector of G is a solution of $M \cdot \overline{z} = \overline{b}$, then it is also a solution of the linear system $M^\dagger \cdot M\overline{z} = M^\dagger \cdot \overline{b}$, where $M^\dagger$ denotes the conjugate transpose of M. Moreover, from Lemma 3.3.5, we have that $M^\dagger \cdot M$ is a square matrix of full rank. Thus, this unique solution of $M^\dagger \cdot M\overline{z} = M^\dagger \cdot b$ can be written in closed form as $(M^\dagger M)^{-1}(M^\dagger \cdot \overline{b})$.

Now, by expressing the inverse $(M^\dagger M)^{-1} = \frac{\text{adj}(M^\dagger M)}{\det(M^\dagger M)}$, where adj refers to the 'adjugate matrix', we get a circuit of size (and bit-complexity) poly(s, d, D) for each G_{ij}, and hence, for $G(\overline{x}, y) = \sum_{i=0}^{d} \left(\sum_{j=0}^{D} G_{ij} \right) \cdot y^i$. □

3.3.2 Irreducible polynomials in the field of fractions of a UFD

Lemma 3.3.4 (Gauss' Lemma (Corollary 6.10 [GG13])). *Let R be a unique factorisation domain with field of fractions K. Suppose a polynomial $f \in R[y]$ is monic in y. Then f is irreducible in $K[y]$ if and only if f is irreducible in $R[y]$.*

As a corollary, the factorisation of any monic polynomial f into its irreducible factors in $R[y]$ is exactly the factorisation of f into its irreducible factors in $K[y]$.

3.3.3 A basic linear algebra fact

We use the following basic fact in the proof of correctness of our algorithm.

Lemma 3.3.5. *Let M be an $r \times c$ matrix with entries in $\mathbb{C}[\overline{x}]$ such that $r \geqslant c$ and the rank of M over the field $\mathbb{C}(\overline{x})$ equals c.*

Then, the rank of the $c \times c$ matrix $M^{\dagger}M$ over the field $\mathbb{C}(\overline{x})$ is also equal to c. Here, $M^{\dagger}$ denotes the conjugate transpose of M.

Proof. Since the rank of M over $\mathbb{C}(\overline{x})$ equals c, there is a substitution $\overline{a} \in \mathbb{C}^n$ for the variables such that $N = M(\overline{a})$ has rank c over $\mathbb{C}$. To show that $M^{\dagger}M$ is full rank over $\mathbb{C}(\overline{x})$ it suffices to show that $N^{\dagger}N$ is full rank over $\mathbb{C}$. We do this by arguing that the kernel of $N^{\dagger}N$ does not contain a non-zero vector.

Let $u \in \mathbb{C}^n$ be an arbitrary non-zero vector. We now argue that any such u cannot be in the kernel of $N^{\dagger}N$. To see this, consider the inner product $\langle u, N^{\dagger}Nu\rangle$ where $\langle u, v\rangle := u^{\dagger}v$. Clearly, this equals $\langle Nu, Nu\rangle$, which equals the square of the ℓ_2 norm of the vector Nu. Since N has full column rank and u us non-zero, we have that Nu is a non-zero vector in $\mathbb{C}^n$, and thus its ℓ_2 norm is non-zero. Thus, $N^{\dagger}Nu$ must be non-zero, and $N^{\dagger}N$ must be full rank. □

3.3.4 Irreducibility under a shift of variables

Since the final step of solving a linear system after Newton iteration works only when the candidate factor is irreducible, we need to ensure that the initial shift of variables does not affect the irreducibility of the factors; this is guaranteed by the following lemma.

Lemma 3.3.6. *Let $\mathbb{F}$ be a field and $g(\overline{x}) \in \mathbb{F}[\overline{x}]$ be an n-variate irreducible polynomial. Then, for every $\overline{a} \in \mathbb{F}^n$, the polynomial $G(\overline{x}, y) := g(\overline{x} + \overline{a} \cdot y)$ is also irreducible.*

Proof. We will interpret the polynomial $g(\overline{x}) \in \mathbb{F}[\overline{x}]$ as $g(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ with the property that it does not depend on the variable y. The mapping $(\overline{x}, y) \mapsto (\overline{x} + \alpha \cdot y, y)$ is an invertible linear transformation M on the vector of variables $(x_1, \ldots, x_n, y) = (\overline{x}, y)$.

We shall prove the contrapositive of the claim: if the polynomial $G(\overline{x}, y) := g(M(\overline{x}, y))$ has a non-trivial factorisation, then so does g. Formally, if $g(M(\overline{x}, y)) = g_1(\overline{x}, y) \cdot g_2(\overline{x}, y)$ for some non-constant polynomials g_1 and g_2, then $g(\overline{x}, y) = g_1(M^{-1}(\overline{x}, y)) \cdot g_2(M^{-1}(\overline{x}, y))$ such that $g_1(M^{-1}(\overline{x}, y))$ and $g_2(M^{-1}(\overline{x}, y))$ are also non-constant polynomials. Observe that the operation of taking a product of two polynomials commutes with the operation of applying a linear transformation on the variables; it is easy to see for monomials, and it follows for polynomials by linearity of M. Thus, applying M^{-1} on both sides of $g(M[\overline{x}, y]) = g_1(\overline{x}, y) \cdot g_2(\overline{x}, y)$ gives us that $g(\overline{x}, y) = g_1(M^{-1}(\overline{x}, y)) \cdot$

$g_2(M^{-1}(\overline{x}, y))$. Here, $g_1(M^{-1}(\overline{x}, y))$ and $g_2(M^{-1}(\overline{x}, y))$ will be non-constants since M is invertible. Thus, if G has a non-trivial factorisation, then so does g. □

3.3.5 Converting a circuit over a number field to a circuit over rational numbers

Lemma 3.3.7 (Constructing a circuit over the base field)**.** *Suppose we are given as input an irreducible polynomial $A(u)$ of degree r and bit-complexity B, and a circuit C, with divisions, of size and formal-degree at most s over a field $\frac{\mathbb{Q}[u]}{A(u)}$ that computes a polynomial $g(\overline{x}) \in \mathbb{Q}[\overline{x}]$. Further, the formal degree of u (interpreted as a variable) in C is bounded by d. Then, there is a deterministic algorithm that can output another circuit C' with size* $\mathrm{poly}(s, r, d, B)$ *over the field $\mathbb{Q}$ computing the polynomial $g(\overline{x})$.*

Proof. For any polynomial $p(u) \in \mathbb{Q}[u]$, we will use $(p \bmod A(u))$ to denote the unique polynomial $p'(u)$ of degree less than $A(u)$ such that $A(u) \mid p(u) - p'(u)$.

We may assume without loss of generality[3] that the input circuit C only has a division gate at the root, i.e. it computes g in the form $g(\overline{x}) = \frac{C_{\mathrm{num}}(\overline{x})}{C_{\mathrm{den}}(\overline{x})}$ where C_{num} and C_{den} are circuits without divisions over $\mathbb{Q}[u]/A(u)$. Let s, d be bounds on the size and degree of the circuits, respectively, of C_{num} and C_{den}.

By interpreting u as a formal variable, let $C'_{\mathrm{num}}, C'_{\mathrm{den}} \in \mathbb{Q}[\overline{x}, u]$ be the resulting polynomials (so that $C'_{\mathrm{num}} \bmod A(u) = C_{\mathrm{num}}$ and similarly for C'_{den}). We may write $C'_{\mathrm{num}}, C'_{\mathrm{den}}$ as

$$\begin{aligned}
C'_{\mathrm{num}} &= C'^{(0)}_{\mathrm{num}} + C'^{(1)}_{\mathrm{num}} u + \cdots + C'^{(d)}_{\mathrm{num}} u^d \in \mathbb{Q}[\overline{x}, u] \\
C'_{\mathrm{den}} &= C'^{(0)}_{\mathrm{den}} + C'^{(1)}_{\mathrm{den}} u + \cdots + C'^{(d)}_{\mathrm{den}} u^d \in \mathbb{Q}[\overline{x}, u]. \\
C_{\mathrm{num}} = (C'_{\mathrm{num}} \bmod A(u)) &= C^{(0)}_{\mathrm{num}} + C^{(1)}_{\mathrm{num}} u + \cdots + C^{(r-1)}_{\mathrm{num}} u^{r-1} \\
&= \sum_{i=0}^{d} C'^{(0)}_{\mathrm{num}} (u^i \bmod A(u)) \\
C_{\mathrm{den}} = (C'_{\mathrm{den}} \bmod A(u)) &= C^{(0)}_{\mathrm{den}} + C^{(1)}_{\mathrm{den}} u + \cdots + C^{(r-1)}_{\mathrm{den}} u^{r-1} \\
&= \sum_{i=0}^{d} C'^{(0)}_{\mathrm{den}} (u^i \bmod A(u))
\end{aligned}$$

Using Lemma 2.3.6, we can obtain circuits for each $C'^{(i)}_{\mathrm{den}}, C'^{(i)}_{\mathrm{den}}$ of size at most $\mathrm{poly}(s, d)$, and thus we can compute circuits for $C^{(i)}_{\mathrm{num}}$ and $C^{(i)}_{\mathrm{den}}$ as well.

3 The standard 'division-elimination' approach keeps tracks of numerators and denominators in gate of the original circuit. This transformation only increases the size by a polynomial factor.

To 'invert' C_{den}, we consider indeterminates $b_0, b_1, \ldots, b_{r-1}$ such that

$$(C_{\text{den}}^{(0)} + \cdots + C_{\text{den}}^{(r-1)} u^{r-1}) \cdot (b_0 + b_1 u + \cdots + b_{r-1} u^{r-1}) = 1 \bmod A(u),$$

which is once again a linear system over the indeterminate b_j's, with coefficients being linear combinations of the circuits $C_{\text{den}}^{(i)}$'s. We can therefore express the b_j's as efficient rational functions using the circuits $C_{\text{den}}^{(i)}$'s via Cramer's rule.

Finally, if $\frac{C_{\text{num}}}{C_{\text{den}}}$ was indeed a polynomial $g(\overline{x}) \in \mathbb{Q}[\overline{x}]$, we must have that

$$((C_{\text{num}} \cdot (b_0 + b_1 u + \cdots b_{r-1} u^{r-1})) \bmod A(u)) = g(\overline{x}).$$

Thus, we can once again compute the LHS above and return the circuit for the coefficient of u^0 in the above expression. The size and bit-complexity bounds follow readily from the description. □

3.4 PSEUDO-RESULTANT

In this section, we will define the *pseudo-resultant* of a polynomial f and a factor g. The pseudo-resultant serves as a proxy for the resultant as it is easy to argue about its circuit complexity, and it suffices for our applications. This definition and its applications to polynomial factorisation that we discuss later in the chapter are essentially the main technical contributions of this work.

As discussed in the proof overview, a way to motivate this definition is to think about the requirement for Newton iteration (Lemma 3.3.1). The simplest setting to keep in mind is when $f(\overline{x}, y) = g(\overline{x}, y)h(\overline{x}, y)$, g is irreducible, $g \nmid h$ and all the polynomials are monic. We want to find some $\overline{a}$ such that there is a root of $f(\overline{a}, y)$ (in particular, a root of $g(\overline{a}, y)$, the univariate "projection" of the factor we care about), which is *not* a root of $\partial_y f(\overline{a}, y)$. If we apply the product rule on $f(\overline{a}, y)$, we get $\partial_y f(\overline{a}, y) = \partial_y g(\overline{a}, y)h(\overline{a}, y) + g(\overline{a}, y)\partial_y h(\overline{a}, y)$. If u is a root of $g(\overline{a}, y)$, it kills the second summand of the derivative, so to ensure $\partial_y f(\overline{a}, u) \neq 0$, we need $\partial_y g(\overline{a}, u) \neq 0$ and $h(\overline{a}, u) \neq 0$. For the latter, it would be enough to ensure that $g(\overline{a}, y)$ has a root which is not a root of $h(\overline{a}, y)$ i.e. $g(\overline{a}, y) \nmid h(\overline{a}, y)$; this is a divisibility-testing condition and can be turned into a PIT using the pseudo-quotient (Theorem 2.3.18). The first condition – $\partial_y g(\overline{a}, u) \neq 0$ – requires that we avoid u that are roots of multiplicity at least 2 for $g(\overline{a}, y)$ since these will also be roots for the first derivative. The pseudo-resultant will be constructed to satisfy the two requirements. We shall now go ahead with the formal definition.

For a polynomial $f(\overline{x}, y)$, let $S_f(\overline{x}, y)$ denote $(\partial_y f(\overline{x}, y))^2$ and $S_{f(\overline{a}, y)}(y)$ denotes $(\partial_y f(\overline{a}, y))^2$.

Definition 3.4.1 (Pseudo-resultant). *Given* $f, g \in \mathbb{F}[\overline{x}, y]$, *let* $Q(\overline{x}, y)$ *be the pseudo-quotient of* S_f *and* g *interpreted as univariates in* y *(with coefficients in* $\mathbb{F}(\overline{x})$*). Then, the* pseudo-resultant *of* f *and* g *with respect to* y*, denoted by* $R_{f,g}$*, is a polynomial in* $\mathbb{F}[\overline{x}, y]$ *defined as*

$$R_{f,g} := S_f - Q \cdot g. \qquad \Diamond$$

The variable y is special in the above definition, and will remain so throughout the chapter.

We make the following simple observation that summarizes a natural property of the pseudo-quotient and the pseudo-resultant under substitutions.

Observation 3.4.2 (Pseudo-quotients and pseudo-resultants under substitutions). *Suppose* $f(\overline{x}, y), g(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ *are monic in* y*, and suppose* $\overline{a} \in \mathbb{F}^{|\overline{x}|}$ *such that* $g(\overline{a}, 0) \neq 0$*. Let* $Q(\overline{x}, y)$ *be the pseudo-quotient of* S_f *and* g *interpreted as univariates in* y *(with coefficients in* $\mathbb{F}(\overline{x})$*). Let* $Q'(y)$ *be the pseudo-quotient of* $S_{f(\overline{a},y)}$ *and* $g(\overline{a}, y)$*. Let* $R_{f(\overline{a},y),g(\overline{a},y)} := S_{f(\overline{a},y)} - Q' \cdot g(\overline{a}, y)$ *be the pseudo-resultant of* $f(\overline{a}, y)$ *and* $g(\overline{a}, y)$*. Then:*

1. $Q(\overline{a}, y) = Q'(y)$.
2. $R_{f,g}(\overline{a}, y) = R_{f(\overline{a},y),g(\overline{a},y)}(y)$.

Proof. For the first part, let $g(\overline{x}, y) = g_0(\overline{x}) + g_1(\overline{x})y + \cdots + y^d$. By the definition of the pseudo-quotient, we have

$$Q(\overline{x}, y) = \mathrm{Hom}_{\leqslant D-d}\left[\frac{S_f(\overline{x}, y)}{g_0(\overline{x})}\left(1 + \tilde{g}(\overline{x}, y) + \tilde{g}(\overline{x}, y)^2 + \cdots + \tilde{g}(\overline{x}, y)^{D-d}\right)\right],$$
$$Q'(y) = \mathrm{Hom}_{\leqslant D-d}\left[\frac{S_{f(a,y)}(y)}{g_0(\overline{a})}\left(1 + \hat{g}(y) + \hat{g}(y)^2 + \cdots + \hat{g}(y)^{D-d}\right)\right]$$

where $\tilde{g}(\overline{x}, y) = 1 - \frac{g(\overline{x},y)}{g_0(\overline{x})}$ and $\hat{g}(y) = 1 - \frac{g(\overline{a},y)}{g_0(\overline{a})}$. It is evident that $\hat{g}(y) = \tilde{g}(\overline{a}, y)$. Since the operations of taking partial derivatives with respect to y (in the definitions of S_f and $S_{f(\overline{a},y)}$), taking homogeneous components with respect to y and evaluating the $\overline{x}$ variables commute with each other, it follows that $Q(\overline{a}, y) = Q'(y)$ as claimed.

For the same reason, it follows that $S_f(\overline{x}, y) = S_{f(\overline{a},y)}(y)$. Combining this observation with the first part, we get $S_f(\overline{a}, y) - Q(\overline{a}, y)g(\overline{a}, y) = S_{f(\overline{a},y)}(y) - Q'(y)g(\overline{a}, y)$ i.e. $R_{f,g}(\overline{a}, y) = R_{f(\overline{a},y),g(\overline{a},y)}(y)$. □

The following lemma tells us how to use the pseudo-resultant to find a suitable starting point $\overline{a}$ for Newton iteration.

Lemma 3.4.3 (Properties of pseudo-resultant). *Suppose* $\mathbb{F}$ *is a field,* $f \in \mathbb{F}[\overline{x}, y]$ *is monic in* y*, and* $f = g \cdot h$*, where* g *is irreducible and* $g \nmid h$*. Let* $S_f(\overline{x}, y) = (\partial_y f(\overline{x}, y))^2$*, let* $R_{f,g}$ *denote the pseudo-resultant of* f *and* g *with respect to* y*, and let* $R_{f(\overline{a},y),g(\overline{a},y)}(y)$ *denote the pseudo-resultant of* $f(\overline{a}, y)$ *and* $g(\overline{a}, y)$ *with respect to* y*. Then, the following properties hold.*

1. $g \nmid S_f$ *in* $\mathbb{F}[\overline{x}, y]$ *and* $\mathbb{F}(\overline{x})[y]$. *Equivalently,* $R_{f,g}(\overline{x}, y) \not\equiv 0$ *in* $\mathbb{F}(\overline{x})[y]$.

2. *For any* $\overline{a} \in \mathbb{F}^n$ *such that* $R_{f,g}(\overline{a}, y) \not\equiv 0$ *in* $\mathbb{F}[y]$, $\overline{a}$ *satisfies the following property: there exists* $u \in \overline{\mathbb{F}}$ *such that* $f(\overline{a}, u) = 0$, $h(\overline{a}, u) \neq 0$ *and* $\partial_y f(\overline{a}, u) \neq 0$. *In particular,* u *is a multiplicity-1 root of* $f(\overline{a}, y)$.

Proof. Since the polynomial f is monic in y, we have that g, h and S_f are all monic in y as well. Thus, by Gauss' Lemma (Lemma 3.3.4), we get that divisibility in $\mathbb{F}[\overline{x}, y]$ is the same as that in $\mathbb{F}(\overline{x})[y]$, so we focus on divisibility in $\mathbb{F}[\overline{x}, y]$.

From $f = g \cdot h$ and the product rule for derivatives, we get

$$\partial_y f = h \cdot \partial_y g + g \cdot \partial_y h .$$

Now, since g is irreducible, we have that g divides $(\partial_y f)^2$ if and only if g divides $\partial_y f$. This, in turn, happens if and only if g divides $h \cdot \partial_y g$. But from the hypothesis of the lemma, we know that g is irreducible, and does not divide h. Moreover, since the y-degree of $\partial_y g$ is less than the y-degree of g, we have that g does not divide $\partial_y g$ either. Thus, from the irreducibility of g, it follows that g does not divide the product $h \cdot \partial_y g$, and hence, g does not divide $(\partial_y f)^2$ (in the ring $\mathbb{F}[\overline{x}, y]$). As mentioned earlier, it now follows from Gauss' lemma (Lemma 3.3.4) that g does not divide $(\partial_y f)^2$ (in the ring $\mathbb{F}(\overline{x})[y]$) i.e. $g \nmid S_f$. Combining the definition of $R_{f,g}$ and Forbes' reduction of divisibility testing to PIT (Theorem 2.3.18), it follows that $R_{f,g}(\overline{x}, y) \not\equiv 0$ in $\mathbb{F}(\overline{x}, y)$. This completes the proof of the first item.

We now proceed with the proof of the second item. Let $\overline{a}$ be a setting of the $\overline{x}$-variables such that $R_{f,g}(\overline{a}, y) = R_{f(\overline{a},y),g(\overline{a},y)}(y) \not\equiv 0$ (where the equality follows from Observation 3.4.2) or equivalently, $g(\overline{a}, y) \nmid S_f(\overline{a}, y)$ (by Theorem 2.3.18 and the fact that $S_f(\overline{a}, y) = S_{f(\overline{a},y)}(y)$). Let $g(\overline{a}, y)$ factor as $g(\overline{a}, y) = \prod_i (y - \alpha_i)^{e_i}$ over the algebraic closure $\overline{\mathbb{F}}$ of the field $\mathbb{F}$. Let $g_1(y) = \prod_{i:e_i=1} (y - \alpha_i)$ be the product of linear factors of multiplicity 1, and $g_2(y) = \prod_{i:e_i \geqslant 2} (y - \alpha_i)^{e_i}$ be the product of rest of the factors. We first claim that for every root α_i of $g_2(y)$ with multiplicity $e_i \geqslant 2$, it holds that $(y - \alpha_i)^{e_i} \mid (\partial_y g(\overline{a}, y))^2$. Indeed, denote by $q(y)$, the polynomial $g(\overline{a}, y)/(y - \alpha_i)^{e_i}$. Then, by the product rule, we have

$$\partial_y g(\overline{a}, y) = e_i (y - \alpha)^{e_i - 1} q(y) + \partial_y q(y) \cdot (y - \alpha_i)^{e_i} ,$$

Since $e_i \geqslant 2$, we have that $2(e_i - 1) \geqslant e_i$. Thus, $(y - \alpha_i)^{e_i} \mid (\partial_y g(\overline{a}, y))^2$ and more generally, since $(y - \alpha_i)^{e_i}$ and $(y - \alpha_j)^{e_j}$ are relatively prime whenever $\alpha_i \neq \alpha_j$, we get that $g_2(y) \mid (\partial_y g(\overline{a}, y))^2$.

Furthermore, setting $\overline{x}$ to $\overline{a}$ in the expression for $S_f(\overline{x}, y)$, we get that

$$\begin{aligned} S_f(\overline{a}, y) = {} & (\partial_y g(\overline{a}, y) \cdot h(\overline{a}, y))^2 + (g(\overline{a}, y) \cdot \partial_y h(\overline{a}, y))^2 \\ & + 2 \cdot (\partial_y g(\overline{a}, y)) \cdot g(\overline{a}, y) \cdot h(\overline{a}, y) \cdot (\partial_y h(\overline{a}, y)) . \end{aligned}$$

We note here that all the partial derivatives in the above expression are taken with respect to the variable y, whereas all the substitutions are happening with respect to the $\overline{x}$-variables, and thus these two operations commute with each other. From the form of the expression above, the definition of g_2 and our earlier observation that $g_2(y) \mid (\partial_y g(\overline{a}, y))^2$, it immediately follows that $g_2(y) \mid S_f(\overline{a}, y)$.

Since $g(\overline{a}, y)$ does not divide $S_f(\overline{a}, y)$ but $g_2(y)$ does, it must be the case that $g_1(y)$ does not divide $S_f(\overline{a}, y)$. Thus, $g_1(y)$ has degree at least one and at least one of its roots in $\overline{\mathbb{F}}$ is *not* a root of $S_f(\overline{a}, y)$. We claim that this root satisfies the requirements of being the $u \in \overline{\mathbb{F}}$ from the claim. To this end, we note that

- $S_f(\overline{a}, u) \neq 0$ or equivalently, $\partial_y f(\overline{a}, u) \neq 0$
- $f(\overline{a}, u) = g(\overline{a}, u) \cdot h(\overline{a}, u) = g_1(u) \cdot g_2(u) \cdot h(\overline{a}, u) = 0$ since we chose u so that $g_1(u) = 0$
- $\partial_y f(\overline{a}, u) = \partial_g(\overline{a}, u) \cdot h(\overline{a}, u) + \partial_h(\overline{a}, u) \cdot g(\overline{a}, u) = \partial_y g(\overline{a}, u) \cdot h(\overline{a}, u)$ by the previous bullet. But $\partial_y f(\overline{a}, u) \neq 0$ and hence $h(\overline{a}, u) \neq 0$.

This completes the proof of the second item. □

3.4.1 A starting point for Newton iteration

The next lemma proves that in the setting of constant-depth circuits, the complexity of the pseudo-resultant of f and g is comparable to the complexity of f and g. Hence, if we want to find a point $\overline{a}$ that will help us start Newton iteration using the sufficient condition from Lemma 3.4.3 (property 2), then it is enough to consider points in a hitting set for constant-depth circuits.

Lemma 3.4.4. *Suppose $f \in \mathbb{F}[\overline{x}, y]$ is a polynomial of degree D with the following properties.*

- *f is monic in y.*
- *$f = g \cdot h$, where $g = \sum_i g_i(\overline{x}) y^i$ is a monic irreducible polynomial of degree d such that $g \nmid h$, and $g_0(\overline{x}) \neq 0$.*
- *f can be computed by a circuit in $(\Sigma\Pi)^{(\Delta_1)}$ of size at most s_1 and g can be computed by a size s_2 circuit in $(\Sigma\Pi)^{(\Delta_2)}$ for some constants $\Delta_1, \Delta_2 \in \mathbb{N}$.*

Let $R_{f,g}(\overline{x}, y) \in \mathbb{F}(\overline{x})[y]$ be the pseudo-resultant of f and g with respect to y. For $\Delta = \max(\Delta_1, \Delta_2)$, let $\mathcal{H}$ be a hitting set for $(n+1)$-variate $(\Sigma\Pi)^{(\Delta+2)}$-circuits of size at most $11 s_1 s_2 D^5$ and degree at most $5D^2$. Then, there exists a point $\tilde{\overline{a}} = (a_1, \ldots, a_n, a_{n+1}) = (\overline{a}, a_{n+1}) \in \mathcal{H}$ such that $R_{f,g}(\overline{a}, y) \not\equiv 0$.

Proof. Let $Q(\overline{x}, y)$ be the pseudo-quotient of S_f and g as univariates in y. Let the numerator and the denominator of $R_{f,g}(\overline{x}, y) = S_f(\overline{x}, y) - Q(\overline{x}, y) \cdot g(\overline{x}, y)$ be $P_{num}(\overline{x}, y)$ and $P_{den}(\overline{x})$ respectively. The proof proceeds by bounding the complexity of the numerator and the denominator, and observing that the given hitting set suffices to find a non-zero for both of them.

BOUNDING THE COMPLEXITY OF THE DENOMINATOR $P_{den}(\overline{x})$: $S_f(\overline{x}, y)$ and $g(\overline{x}, y)$ do not contribute to the denominator. The only contributions to $P_{den}(\overline{x})$ (from $Q(\overline{x}, y)$) are $g_0(\overline{x})$ by the $\frac{S_f(\overline{x},y)}{g_0(\overline{x})}$ term, and $g_0(\overline{x})^{D-d}$ from the powers of $\tilde{g}$. Thus, $P_{den}(\overline{x}) = g_0(\overline{x})^{D-d+1}$. Since $g(\overline{x}, y)$ has a $(\Sigma\Pi)^{(\Delta_2)}$-circuit of size s_2, so does $g_0(\overline{x})$ (obtained by just setting y to zero). Thus, $P_{den}(\overline{x})$ has a $(\Sigma\Pi)^{(\Delta_2+1)}$-circuit of size at most $s_2 + D \leqslant s_2 D$. The degree of P_{den} is clearly upper bounded by D^2.

BOUNDING THE COMPLEXITY OF THE NUMERATOR $P_{num}(\overline{x}, y)$: The numerator of $Q(\overline{x}, y)$, denoted by $Q_{num}(\overline{x}, y)$, is

$$Q_{num}(\overline{x}, y) = \mathrm{Hom}_{D-d}\left[S_f(\overline{x}, y)\left(\sum_{i=0}^{D-d} g_0^{D-d-i}(g_0 - g)^i\right)\right].$$

From the definition of S_f and standard computation of partial derivatives (see Lemma 2.3.8), $S_f(\overline{x}, y)$ has a $(\Sigma\Pi)^{(\Delta_1)}$-circuit of size $(10 s_1 D^3)$. $\sum_{i=0}^{D-d} g_0^{D-d-i}(g_0 - g)^i$ has a $(\Sigma\Pi)^{(\Delta_2+1)}$-circuit of size $s_2 D^3$. Taking its product with $S_f(\overline{x}, y)$ and using interpolation to get the homogeneous components of degree at most $D - d$ (Lemma 2.3.7) gives us a $(\Sigma\Pi)^{(\Delta+1)}$-circuit of size $(10 s_1 D^5)$ for Q_{num}. Thus, $P_{num}(\overline{x}, y) = P_{den}(\overline{x}) S_f(\overline{x}, y) - Q_{num}(\overline{x}, y) g(\overline{x}, y)$ has a $(\Sigma\Pi)^{(\Delta+1)}$-circuit of size $10 s_1 s_2 D^5$. From the expression of the numerator, we have that a crude bound on its degree is at most $2D + D^2 + D^2 \leqslant 4D^2$.

THE HITTING SET $\mathcal{H}$ SUFFICES : Since $P_{num}(\overline{x}, y)$ and $P_{den}(\overline{x})$ have relatively small $(\Sigma\Pi)^{(\Delta+1)}$-circuits, then their product has a $(\Sigma\Pi)^{(\Delta+2)}$-circuit of size at most $10 s_1 s_2 D^5 + s_2 D \leqslant 11 s_1 s_2 D^5$ and its degree is at most $D^2 + 4D^2 = 5D^2$. Thus, a hitting set $\mathcal{H}$ for this class will contain a point $\tilde{\overline{a}} = (a_1, \ldots, a_n, a_{n+1}) = (\overline{a}, a_{n+1})$ such that $P_{num}(\tilde{\overline{a}}) \cdot P_{den}(\overline{a})$ is non-zero. In particular, $\frac{P_{num}(\overline{a},y)}{P_{den}(\overline{a})} = R_{f,g}(\overline{a}, y) \not\equiv 0$.

□

3.5 GENERATING A LIST OF CANDIDATE FACTORS

Algorithm 3: Computing a list of candidate irreducible factors of multiplicity-1, degree d and computable by a depth Δ' circuit of size m, when the input polynomial has depth Δ

Input : A size parameter m in unary; degree parameter $d \in \mathbb{N}$; a $(\Sigma\Pi)^{\Delta}$-circuit of size s, bit-complexity t, total degree D, computing a polynomial $f(\overline{x}, y) \in \mathbb{Q}[x_1, \ldots, x_n, y]$ that is monic in y.

Output: A list $L = \{C_1, \ldots, C_r\}$ of algebraic circuits (with division gates) such that for every irreducible factor $g(\overline{x}, y)$ of $f(\overline{x}, y)$ s.t. $g^2 \nmid f$ and g has a size m circuit of depth Δ', y-degree equal to d, there exists $i \in [r]$ such that $C_i \equiv g(\overline{x}, y)$.

1 Set the list $L' = \emptyset$.

2 Compute hitting-set H_1 for $(n+1)$-variate $(\Sigma\Pi)^{(\max(\Delta,\Delta')+2)}$-circuits of size $(11smD^5)$ and degree $(5D^2)$ using Theorem 2.3.13.

3 Let H_2 be the projection of the points in H_1 on the first n coordinates.

4 **for** $\overline{a} = (a_1, \ldots, a_n) \in H_2$ **do**

5 $\quad F(\overline{x}, y) := f(x_1 + a_1, \ldots, x_n + a_n, y)$

6 $\quad$ Factorise the polynomial $F(\mathbf{0}, y) \in \mathbb{Q}[y]$ into irreducible factors as

$$F(\mathbf{0}, y) = \sigma \cdot F_1(y)^{e_1} \cdots F_l(y)^{e_l}.$$

$\quad$ where $0 \neq \sigma \in \mathbb{Q}$ and each $F_i(y)$ is monic.

7 $\quad$ **forall** F_i *s.t.* $e_i = 1$ **do**

8 $\quad\quad$ Set $\mathbb{K} := \frac{\mathbb{Q}[u]}{\langle F_i(u) \rangle}$.

9 $\quad\quad$ **if** $\partial_y F(\mathbf{0}, u) = 0$ **then**

10 $\quad\quad\quad$ Skip to the next F_i.

11 $\quad\quad$ Use Lemma 3.3.1 to obtain a circuit for $\Phi_k(\overline{x})$ such that $F(\overline{x}, \Phi_k) = 0 \bmod \langle \overline{x} \rangle^{k+1}$ for $k = 2D^2$.

12 $\quad\quad$ Use Lemma 3.3.3 to obtain a circuit (with divisions) C for the minimal polynomial $G(\overline{x}, y)$ for Φ_k modulo the ideal $\langle \overline{x} \rangle^{k+1}$.

13 $\quad\quad$ Apply the transformation $x_i \to x_i - a_i$ on C to get a circuit C'.

14 $\quad\quad$ Update $L' := L' \cup \{C'\}$.

15 **return** L'

Remark (Obtaining circuits over the base field $\mathbb{Q}$). *Although the above algorithm outputs circuits over algebraic extensions of $\mathbb{Q}$ of the form $\mathbb{K} = \frac{\mathbb{Q}[u]}{A(u)}$, they can be transformed syntactically to circuits over the base field $\mathbb{Q}$, with only a polynomially large blow-up in size, via standard techniques (Lemma 3.3.7).* ◊

For simplicity, Algorithm 3 deals with the special case when the factors we care about have multiplicity one and a fixed degree d. Later, we will describe the final algorithm that will iterate over the values

of the degree and multiplicity parameters, and run Algorithm 3 as a subroutine.

We now bound the running time of the above algorithm, and prove its correctness. In the following subsection, we use this algorithm to prove the main theorem Theorem 1.7.5.

3.5.1 Proof of correctness of Algorithm 3

Theorem 3.5.1 (Correctness of Algorithm 3)**.** *Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $\Delta, \Delta' \in \mathbb{N}$ be arbitrary constants. Let $f(\overline{x}, y) \in \mathbb{Q}[\overline{x}, y]$ be any polynomial that is monic in y, is computed by a $(\Sigma\Pi)^{(\Delta)}$-circuit of size s, bit-complexity B, degree D and let g be an irreducible factor of multiplicity one of f with y-degree exactly d such that g is computable by a $(\Sigma\Pi)^{(\Delta')}$-circuit of size m.*

If Algorithm 3 is invoked on input m, d, a circuit for f with size s, bit-complexity t, D, then the output is a list L of circuits of size $\mathrm{poly}(s, m, D)$ and bit-complexity $\mathrm{poly}(s, t, m, D)$ such that L contains at least one circuit that computes g.

Moreover, the algorithm terminates in time $(O(smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\epsilon((smD^7)^\epsilon)} \cdot \mathrm{poly}(s, D, B)$, where $\tilde{\Delta} = \max(\Delta, \Delta') + 2$.

Proof. We start with the proof of correctness. Suppose $f = g \cdot h$ where g is one of the irreducible factors of interest. Let $R_{f,g}$ be the pseudo-resultant of f and g with respect to y. From Lemma 3.4.4, we get that there exists a point $(a_1, a_2, \ldots, a_n, a_{n+1})$ in the hitting set H_1 defined in the algorithm such that $R_{f,g}(a_1, \ldots, a_n, y) \not\equiv 0$. In other words, there is an $\overline{a} = (a_1, a_2, \ldots, a_n) \in H_2$ such that $R_{f,g}(\overline{a}, y) \not\equiv 0$; let us fix such an $\overline{a}$.

From this and the second item of Lemma 3.4.3, we get that there is a $u \in \overline{\mathbb{Q}}$ such that $g(\overline{a}, u) = 0$, $h(\overline{a}, u) \neq 0$ and $\partial_y(f)(\overline{a}, u) \neq 0$; moreover, this u must be a root of one of the F_i's with multiplicity $e_i = 1$ obtained in Line 6. Hence, $F(\mathbf{o}, u) = f(\overline{a}, u) = 0$ and $\partial_y F(\mathbf{o}, u) = \partial_y f(\overline{a}, u) \neq 0$. We now satisfy the hypothesis for Newton Iteration (Lemma 3.3.1) and we can obtain a circuit for the approximate root Φ_k for F satisfying $\Phi_k(\mathbf{o}) = u$. Let $G(\overline{x}, y) := g(\overline{x} + \overline{a}, y)$ and $H(\overline{x}, y) := h(\overline{x} + \overline{a}, y)$. Since $F(\overline{x}, \Phi_k) = G(\overline{x}, \Phi_k) \cdot H(\overline{x}, \Phi_k) = 0 \bmod \langle \overline{x} \rangle^{k+1}$ and $H(\mathbf{o}, \Phi_k(\mathbf{o})) = h(\overline{a}, u)$ is a nonzero scalar in $\mathbb{K}$, we have that $G(\overline{x}, \Phi_k) = 0 \bmod \langle \overline{x} \rangle^{k+1}$. Therefore, for $k = 2D^2$ (as set in the algorithm), Lemma 3.3.3 will yield a circuit (with divisions) for G. Undoing the initial translate via $x_i \to x_i - a_i$ yields a circuit for $g(\overline{x}, y)$.

From the description of the algorithm, we get that time complexity is at most

$$T_0 + |H_2| \cdot D \cdot (T_1 + T_2 + T_3)$$

where, T_0 is the time taken to computing an appropriate hitting set in Line 2, $T_1 \leqslant \mathrm{poly}(D, B)$ is the time taken to factorise the polynomial $F(\overline{0}, y)$ (Theorem 2.3.27), $T_2 \leqslant \mathrm{poly}(s, D, B)$ is the time

taken for computing the approximate root (Lemma 3.3.1) and computing the minimal polynomial (Lemma 3.3.3). The size of H_2 is at most the size of H_1, and Theorem 2.3.13 tells us that $|H_1| \leqslant ((11smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\epsilon((44smD^7)^\epsilon)}$ (where $\tilde{\Delta} = \max(\Delta, \Delta') + 2$); moreover, Theorem 2.3.13 tells us that T_0, the time-complexity of computing H_1 at Line 2, has the same expression as the size of H_1. Thus, the total running time is $((11smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\epsilon((55smD^7)^\epsilon)} \cdot \mathrm{poly}(s, D, B)$. The bit-complexity bound follows as the only nontrivial constant added to each circuit is the algebraic number u whose minimal polynomial is one of the factors of $F(\overline{0}, y)$ and hence has small bit-complexity as well. □

3.5.2 Proof of Theorem 1.7.5

We now describe our final algorithm, and its analysis to complete the proof of Theorem 1.7.5.

Algorithm 4: Computing a list of candidate irreducible factors computable by a depth Δ' circuit of size m, when the input polynomial has depth Δ

Input : A size parameter m in unary; a $(\Sigma\Pi)^{\Delta}$-circuit of size s, bit-complexity t, degree D, computing a polynomial $f(\overline{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$.

Output: A list $L = \{C_1, \ldots, C_r\}$ of algebraic circuits (with division gates) such that for every irreducible factor $g(\overline{x})$ of $f(\overline{x})$ s.t. g has a size m circuit of depth Δ', there exists $i \in [r]$ such that $C_i \equiv g(\overline{x})$.

1 Set the output list $L' = \emptyset$.
2 Compute hitting-set H_1 for n-variate $(\Sigma\Pi)^{(\Delta)}$-circuits of size (sD^3) and degree D using Theorem 2.3.13.
3 **for** $\overline{\alpha} \in H_1$ **do**
4 Define $\tilde{f}(\overline{x}, y) := f(\overline{x} + \overline{\alpha} \cdot y) = f(x_1 + \alpha_1 y, \ldots, x_n + \alpha_n y)$
5 **for** $i = 0 \ldots D - 1$ **do**
6 Compute a circuit C_i for the polynomial $\tilde{f}_i(\overline{x}, y) := \frac{\partial^i \tilde{f}}{\partial y^i}$
7 **for** $d = 1 \ldots D - 1$ **do**
8 Compute a list $L_{i,d}$ of candidate irreducible factors of $\tilde{f}_i(\overline{x}, y)$, with y-degree d and multiplicity one, using Algorithm 3
9 Set $L' := L' \cup L_{i,d}$
10 Let L be the list of circuits obtained by setting y to 0 in circuits in L'.
11 **return** L

Theorem 3.5.2 (Correctness of Algorithm 4)**.** *Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $\Delta, \Delta' \in \mathbb{N}$ be arbitrary constants. Let $f(\overline{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$ be a polynomial computed by a $(\Sigma\Pi)^{(\Delta)}$-circuit of*

size s*, bit-complexity* t*, degree* D *and let* g *be an irreducible factor of* f *such that* g *is computable by a* $(\Sigma\Pi)^{(\Delta')}$*-circuit of size* m*.*

If Algorithm 4 is invoked on input m*, a circuit for* f *with size* s*, bit-complexity* t*,* D*, then the output is a list* L *of circuits of size* $\mathrm{poly}(s, m, D)$ *and bit-complexity* $\mathrm{poly}(s, t, m, D)$ *such that* L *contains at least one circuit that computes* g*. Moreover, the algorithm terminates in time* $\left(((O(smD^8))^{O(\tilde{\Delta})} \cdot n)\right)^{O_\varepsilon(smD^{10})^\varepsilon} \cdot \mathrm{poly}(s, D, T)$*, where* $\tilde{\Delta} = \max\{\Delta, \Delta'\} + 2$*.*

Proof. Let D be the total degree of f. If we view $f(\overline{x} + \overline{\alpha} \cdot y)$ as a formal polynomial in y, we get that the coefficient of y^D in it is equal to the evaluation of the degree D homogeneous component of f on input $(\overline{\alpha})$. Moreover, from Lemma 2.3.6, we get that this homogeneous component has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size at most sD^3. Therefore, there exists an $\overline{\alpha}$ in the hitting set H_1 for which the polynomial $\tilde{f}(\overline{x}, y) = f(\overline{x} + \overline{\alpha} \cdot y)$ has y degree equal to D, and hence (up to multiplication by a non-zero field constant) is monic in y.

We will focus on an arbitrary fixed irreducible factor $g \mid f$ and prove that there exists a circuit in the final output list L that computes g. Let i^* be the multiplicity of g in f. Let the degree of g be d. Then, $\tilde{g}(\overline{x}, y) = g(\overline{x} + \overline{\alpha} \cdot y)$ will be a factor of y-degree-d and multiplicity one of $\tilde{f}_{i^*-1}(\overline{x}, y) := \frac{\partial^i \tilde{f}}{\partial y^i}$. By Lemma 2.3.8, $\tilde{f}_{i^*-1}(\overline{x}, y)$ has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size at most sD^3. Moreover, the transformation $\overline{x} \mapsto \overline{x} + \overline{\alpha} \cdot y$ maintains the irreducibility of g by Lemma 3.3.6. Thus, in the $(i^* - 1)^{th}$ iteration of Line 5 and the dth iteration of Line 7, Line 8 will compute a list of candidate factors that will include $\tilde{g}$ as guaranteed by Theorem 3.5.1. Setting $y = 0$ (which can be done because Lemma 3.3.3 guarantees that only the $\overline{x}$ variables show up in the denominators) to go from the list L' to L will give us a list that contains $\tilde{g}(\overline{x}, 0) = g$.

From the description of the algorithm, the running time of Algorithm 4 is at most

$$T_0 + |H_1| \cdot D^2 \cdot T_1 \cdot \mathrm{poly}(s, D, t)$$

where T_0 is the time taken to compute the hitting set H_1, T_1 is the time taken to invoke Algorithm 3 in Line 8, H_1 is the hitting set computed in Line 2 using Theorem 2.3.13, the D^2 factor is to account for the two loops in Line 5 and Line 7, and the $\mathrm{poly}(s, D, t)$ factor is to account for the rest of the steps such as computing $\tilde{f}$ from f, computing circuits for derivatives, obtaining a list of circuits by setting $y = 0$, etc. T_1 is the running time of the first algorithm on $\tilde{f}_{i^*-1}$ which has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size $s' = sD^3$. Thus, from Theorem 3.5.1, T_1 is at most $((11smD^8)^{O(\tilde{\Delta})} \cdot n)^{O_\varepsilon((55smD^{10})^\varepsilon)} \cdot \mathrm{poly}(s, D, t)$. Applying Theorem 2.3.13 for a $(\Sigma\Pi)^{(\Delta)}$-circuit of size sD^3 and degree D tells us that $((sD^3)^{O(\Delta)} \cdot n)^{O_\varepsilon((sD^4)^\varepsilon)}$ will be a bound on both T_0 as well as the size of H_1. Thus, the final time complexity is

$\left((11smD^8)^{O(\bar{\Delta})} \cdot n\right)^{O_\varepsilon(55smD^{10})^\varepsilon} \cdot \mathrm{poly}(s, D, T)$. The bit-complexity bound follows from Theorem 3.5.1. $\square$

Part II

RECENT HISTORY: AFTER ANDREWS-WIGDERSON, BEFORE FURSTENBERG

Then came those who saw the way,
Through resultants' constant-depth embrace,
Where conjugates speak their kinship true,
Yet closure stood beyond our reach.

4 LIST-FACTORING, BUT BETTER

4.1 OVERVIEW

In this brief chapter, we will sketch out a high-level overview of the algorithm for the following theorem, which we mentioned in the previous chapter.

Theorem 1.7.6. *[Polynomial-sized list containing constant-depth factors of constant-depth circuits, in subexponential time] Let $\mathbb{Q}$ be the field of rational numbers and $\epsilon > 0$, $\Delta \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta)}$ of size s, bit-complexity t and degree D, and outputs a list of circuits $L = \{C_1, \ldots, C_r\}$ with the following properties.

1. *Each $C_i \in L$ is an arithmetic circuit of size $\mathrm{poly}(s, D)$ and bit-complexity $\mathrm{poly}(t, s, D)$, with division gates.*
2. *For every irreducible factor g of C, there exists a $C_i \in L$ such that C_i computes g.*

The size of the list L is $\mathrm{poly}(d)$. The running time of the algorithm is bounded above by $O(sDt)^{O((sDt)^{\epsilon})}$.

While this theorem still has some of the caveats of Theorem 1.7.5 in Chapter 3 – namely, circuits with division gates, and a list of factors with some spurious factors–, there are also significant improvements. The output list of the new algorithm contains representatives for *all* the factors of the input polynomial, not just those factors with constant-depth circuits.[1] Moreover, the list is only polynomially large, as opposed to the algorithm in Theorem 1.7.5, where the list could be subexponentially large.

4.2 PRELIMINARIES

We briefly recall some definitions and theorems relevant to the algorithm.

1 At the time of this result, we did not know that there would eventually be a closure theorem for constant-depth circuits.

4.2.1 Squarefree decomposition

We say that a polynomial P is *squarefree* if it is not divisible by the square of another polynomial. In particular, every irreducible factor of P appears with multiplicity one in the unique factorisation of P.

We now define the notion of squarefree decomposition of a polynomial.

Definition 4.2.1 (Squarefree decomposition)**.** *Let $F \in \mathbb{F}[\overline{x}]$ be a polynomial such that $F(\overline{x}) = \prod_{i=1}^{m} G_i(\overline{x})^{e_i}$. Let $r = \max_{i \in [m]} e_i$, where each G_i is irreducible. Then the squarefree decomposition of F is $(F_1, F_2, \ldots, F_r)$, where for each $i \in [r]$, $F_i := \prod_{j \in [m]: e_j = i} G_j$.* ◊

The following theorem of Andrews & Wigderson shows that squarefree parts of a polynomial computable by a small constant-depth circuit have small constant-depth circuits, and moreover, we can compute such a decomposition deterministically given an appropriate PIT oracle.

Theorem 4.2.2 (Squarefree decomposition [AW24])**.** *Let $\mathbb{F}$ be a field of characteristic zero or characteristic greater than D. Let $\mathcal{O}$ be an oracle that solves polynomial identity testing for constant-depth circuits. Then, there is a deterministic polynomial-time algorithm with oracle access to $\mathcal{O}$ which does the following:*

1. *The algorithm receives as a input a constant-depth circuit that computes a polynomial F of degree D.*
2. *The algorithm outputs a collection of constant-depth circuits $C_1, \ldots, C_r$ such that C_i computes F_i, where $(F_1, \ldots, F_r)$ is the squarefree decomposition of F.*

Remark 4.2.3. *The model of constant depth algebraic circuits as defined in [AW24] are allowed to have division gates. However, an inspection of their proof of Theorem 4.2.2 reveals that an oracle access to a PIT algorithm for division free constant depth algebraic circuits is sufficient to obtain an efficient algorithm that takes as input a constant-depth circuit (without divisions) and outputs constant depth algebraic circuits (again, without division gates) for its squarefree decomposition.*

Moreover, the same argument holds for algebraic formulas, provided we have a PIT oracle for such formulas. ◊

4.3 HIGH-LEVEL SKETCH OF THE ALGORITHM

We now give a high-level sketch of the improved algorithm.

1. Suppose $P(\overline{x})$ is the polynomial that we would like to factor. We first use the algorithm by Andrews and Wigderson (Theorem 4.2.2) to compute the squarefree decomposition of P. Now, we deal with each squarefree part separately.

2. For a specific squarefree part $P_r(\overline{x})$, we perform some of the standard transformations ($x_i \mapsto T \cdot x_i + a_i \cdot z + b_i$) so that the polynomial $P_r(T, \overline{x}, z)$ is monic in z variable, and $P_r(0, \overline{x}, z) = P_r(0, \overline{0}, z)$ is squarefree. The right $\overline{a} = (a_1, \ldots, a_n)$ and $\overline{b} = (b_1, \ldots, b_n)$ can be computed applying an appropriate hitting-set generator for constant-depth circuits (via [CKS19] and [LST25]) on the discriminant of P_r as well as the highest degree homogeneous part of P_r.

3. We apply Newton iteration on each root of $P_r(0, \overline{0}, z)$. But some factors $G(T, \overline{x}, z)$ of $P_r(T, \overline{x}, z)$ might only contribute non-linear irreducible factors in $P_r(0, \overline{0}, z)$. In this situation, we artificially *add a root* u of $G(0, \overline{0}, z)$ to the field; more precisely, if $H(z)$ is an arbitrary irreducible factor of $P_r(0, \overline{0}, z)$, we work over the field $\frac{\mathbb{Q}[u]}{H(u)}$ so that u is now a root of $P_r(0, \overline{0}, z)$. We can efficiently simulate arithmetic in this field. Thus, we can lift the root u to a truncated, approximate z-root $\varphi(T, \overline{x})$ of $P_r(T, \overline{x}, z)$ of sufficiently high accuracy.

4. Given a truncated, approximate z-root φ of $P_r(T, \overline{x}, z)$, we set up a linear system whose solution will give us a minimal polynomial of the root φ. To set up the linear system in a way that ensures that the solution is the same as the right irreducible factor $G(T, \overline{x}, z)$ that has the approximate root φ, we would need to know the exact z-degree of G. Since we do not know the exact z-degree of G, we try all the possibilities and add each solution to the output list. For each degree possibility, the solution to the linear system can be represented as a small circuit using Cramer's rule, and this step introduces a division by a determinant.

Proof sketch for Theorem 1.7.6. The proof of correctness essentially follows along the lines of the proof of Theorem 1.7.5. The main differences are in the size of the output list, and the fact that we can now output a circuit for every factor, not just constant-depth factors. In the proof of Theorem 1.7.5, for each constant-depth factor g of f, we had to use a shift $\overline{\alpha}_g$ that maintains the nonzeroness of the pseudo-resultant of f and g, so that there is some root of the univariate $f(\overline{\alpha}_g, y)$ that satisfies the conditions for starting Newton iteration and recovering g. With no access to the true factor g, the simplest way out was to iterate over the whole hitting set and argue that for each factor g, some point in the hitting set would work. Moreover, for a factor g without a constant-depth circuit, this approach wouldn't work.

Now, [AW24] gives us a squarefree decomposition of the input polynomial. For each output of the decomposition, its Discriminant will be non-trivial (due to squarefreeness), and we can find a point in the hitting-set to maintain this Discriminant. By maintaining squarefreeness, we ensure the conditions required for Newton iteration for

each factor of each output of the squarefree decomposition separately. Thus, there is no need to repeat the rest of the algorithm for *all* choices of shifts from the hitting-set. Moreover, this approach is agnostic of the depth of specific factors.

The reason we can still get spurious factors is because of the fourth steps in the above overview: we do not know the exact z-degree of the right factor $G(T, \overline{x}, z)$, which is required to setup a linear system for recovering G. We can still try all possibilities in $\{1, 2, \ldots, d\}$ for it and setup the linear system in each case. We cannot prune out the spurious factors because they need not have constant-depth circuits. There will also be potentially many copies of each correct factor $G(T, \overline{x}, z)$ since each root or irreducible factor of the $G(0, \overline{0}, z)$ will give us a potentially different unbounded-depth circuit for G, and we will not be able to use a hitting-set generator for constant-depth circuits to remove these duplicate circuits. Taking both these cases into account, the output list will have size at most d^3 — there at most d factors, at most d roots for each factor, and at most d outputs from d linear systems per root. $\square$

5 FACTORS – AND NOTHING ELSE

5.1 OVERVIEW

Our main result in this chapter is a deterministic polynomial factorisation algorithm for constant-depth circuits, as informally stated in the following theorem from Chapter 1. A more formal version is in Theorem 5.5.2.

Theorem 1.7.7. *[Deterministic subexponential time algorithm for factorising constant-depth circuits] For every constant $\epsilon > 0$, constant $\Delta \in \mathbb{N}$ and sufficiently large n, the following statement is true.*

There is a deterministic algorithm that takes as input an algebraic circuit C over the field $\mathbb{Q}$ on n variables with depth Δ, and size and degree $\mathrm{poly}(n)$, runs in time $\exp(O(n^\epsilon))$ and outputs algebraic circuits (of potentially unbounded depth) for all irreducible factors of C, together with their multiplicities.

This result in particular also gives the first subexponential deterministic factoring algorithm for sparse polynomials (which is the case when $\Delta = 2$).

At the heart of our proof is the deterministic construction of a variable reduction map that reduces the number of variables in a constant-depth circuit substantially, while preserving its *factorisation profile*. The following theorem is our main technical result.[1]

Theorem 5.1.1 (Informal version of Theorem 5.5.1)**.** *For every $\epsilon > 0$, every constant $\Delta \in \mathbb{N}$ and all sufficiently large n, there is a polynomial map $\Gamma_{\epsilon,\Delta} : \mathbb{Q}^{n^\epsilon} \to \mathbb{Q}^n$ of degree $o(\log n)$ with the following properties.*

- *Given ϵ, Δ, the map $\Gamma_{\epsilon,\Delta}$ can be constructed deterministically in time $\exp(O(n^\epsilon))$.*
- *$\Gamma_{\epsilon,\Delta}$ is a variable reduction map that preserves the irreducibility of n-variate polynomials that can be computed by depth Δ circuits of size and degree $\mathrm{poly}(n)$. In other words, an n-variate algebraic circuit C of size and degree $\mathrm{poly}(n)$ is irreducible if and only if the n^ϵ-variate polynomial $\hat{C}$ obtained from C by composing it with $\Gamma_{\epsilon,\Delta}$ is irreducible.*

1 The theorem follows immediately from Theorem 5.5.1 and our deterministic implementation of some (fairly standard) preprocessing steps in the factorisation algorithm. In fact, by a slight modification of our proofs, a stronger version of this theorem can be shown to be true where the time complexity of constructing the map $\Gamma_{\epsilon,\Delta}$ is quasipolynomially bounded in n. However, this does not improve the overall time complexity of the algorithm in Theorem 1.7.7 since some steps in the algorithm, for instance, that of deterministic factorisation of n^ϵ variate polynomials of degree $\mathrm{poly}(n)$, still run in time $\exp(O(n^\epsilon))$.

- *More generally, $\Gamma_{\epsilon,\Delta}$ is a variable reduction map that preserves the irreducibility of the factors of n-variate polynomials that can be computed by depth Δ circuits of size and degree* $\mathrm{poly}(n)$.

OBTAINING THEOREM 1.7.7 FROM THEOREM 5.1.1. The second item of the above theorem, when combined with known deterministic factorisation algorithms for n^{ϵ}-variate polynomials of degree at most $\mathrm{poly}(n)$ that run in time $n^{O(n^{\epsilon})}$ (polynomial in the size of the dense representation of such polynomials), essentially gives us a subexponential time deterministic irreducibility testing algorithm for sparse polynomials, and more generally for constant-depth circuits. Similarly, the third item of Theorem 5.1.1 can be combined with known ideas in algorithms for polynomial factorisation (with some effort) to give Theorem 1.7.7.

We end this section with a brief discussion of some quantitative and qualitative aspects of the main theorems.

IMPROVEMENTS IN TIME COMPLEXITY. Given the reduction from polynomial identity testing to polynomial factorisation in [SV10], we know that one cannot hope to improve the running time of the algorithm in Theorem 1.7.7 significantly for general constant-depth circuits, unless we have significantly better deterministic polynomial identity testing algorithms for these problems. However, for the case of sparse polynomials, we have known polynomial time PIT algorithms for more than three decades [BT88; GK87; GKS90; Cla+91; KS01], and thus we can, in principle, expect faster factorisation or irreducibility testing algorithms for sparse polynomials. It is a very natural question to explore further.

CLOSURE RESULTS FOR CONSTANT-DEPTH CIRCUITS. We do not know if irreducible factors of polynomials with small constant-depth circuits (or even the more restricted class of sparse polynomials) must be computable by small constant-depth circuits, and Theorem 1.7.7 does not appear to shine any light on such a closure result. As a consequence, the outputs of the algorithm in Theorem 1.7.7 are general unbounded depth circuits of polynomial size. In fact, we do not even know of non-trivial upper bounds on the size of constant-depth circuits for irreducible factors of sparse polynomials. It would be very interesting to obtain such closure results or to gather some evidence that points towards such a statement being false.

FIELD DEPENDENCE IN THEOREM 1.7.7. Our results in Theorem 1.7.7 are stated over the field of rational numbers. However, the results hold a little more generally - our proofs continue to work over any underlying field over which we have an efficient deterministic algorithm for factoring univariate polynomials, over which the

lower bounds of Limaye, Srinivasan & Tavenas [LST25] for constant-depth circuits, the results of Andrews & Wigderson [AW24], and the Taylor-expansion-based techniques of Chou, Kumar & Solomon [CKS19] continue to work. The field of rational numbers satisfies all these properties. The results also continue to hold over finite fields of moderately large characteristic (polynomially large in the degree and number of variables) since they satisfy all the properties stated above. However, for our presentation in this chapter, we just work over the field of rational numbers and skip the minor technical changes needed to see the extension of the results to finite fields of moderately large characteristic.

5.2 OVERVIEW OF PROOFS

Almost all factorisation algorithms proceed with some initial preprocessing to guarantee the following requirements:

- $P(\overline{x}, z)$ is squarefree and monic in z.
- We have that $P(\overline{0}, 0) = 0$ and $\partial_z(P)(\overline{0}, 0) \neq 0$.

The above non-degeneracy conditions can typically be guaranteed relatively easily via a randomized algorithm. To do this deterministically, we use a recent result of Andrews & Wigderson [AW24] (see Theorem 4.2.2) to get a squarefree decomposition of P. In fact, their theorem says something more that is useful for us: the squarefree parts of P are computable by a small constant-depth circuit since P is computable by a small constant-depth circuit. Moreover, we can obtain this decomposition via a deterministic subexponential time algorithm that uses the now known deterministic PIT algorithms for constant-depth circuits [LST25; AF22].

We also translate the $\overline{x}$ variables appropriately (again using deterministic PIT for constant-depth circuits) to ensure that we are working with polynomials that are monic in z and each of the squarefree parts remains squarefree even when all the $\overline{x}$ variables are set to zero.

We then invoke a univariate factorisation algorithm to find a root of $P(\overline{0}, z)$, which for this discussion we assume is zero. Furthermore, we replace each x_i with $x_i T$, for a fresh variable T, to work with a polynomial of the form $P(T, \overline{x}, z)$. Note that $P(0, \overline{x}, z) \in \mathbb{F}[z]$; we will refer to such polynomials as T-regularized.[2]

The general plan is to use Newton iteration, starting with $\phi_0(T, \overline{x}) = 0$, to get "approximate roots" $\phi_{k-1}(T, \overline{x})$ satisfying $\deg_T(\phi_{k-1}) < k$ and $\phi_{k-1}(T, \overline{x}) = \phi_0(T, \overline{x}) \bmod T$ such that $P(T, \overline{x}, \phi_{k-1}(T, \overline{x})) = 0 \bmod T^k$. We can obtain a small circuit for ϕ_{k-1}, although not nec-

2 More generally, we say that a polynomial in $\mathbb{F}[T, \overline{x}, z]$ is T-regularized if every monomial with $\overline{x}$-degree at least one in the polynomial has T-degree at least one.

essarily one of constant depth. The hope is that, if k is large enough, then we can recover from ϕ_{k-1} a true factor of $P(T, \overline{x}, z)$.

5.2.1 Restricting to roots

Let us start by focusing on just extracting factors of $P(T, \overline{x}, z)$ of the form $(z - f(T, \overline{x}))$. If $f(T, \overline{x}) = \phi_0(T, \overline{x}) \bmod T$, then uniqueness of Newton Iteration guarantees that $\phi_{k-1}(T, \overline{x}) = f(T, \overline{x}) \bmod T^k$ as well. This therefore leads to the following natural algorithm:

1. Compute a circuit for $\phi_{k-1}(T, \overline{x})$ for $k > \deg_T(P)$, with $\deg_T(\phi_{k-1}) < k$, via Newton Iteration.
2. Check if $P(T, \overline{x}, \phi_{k-1}(T, \overline{x})) = 0$.

If it were the case that ϕ_{k-1} was actually computable by a constant-depth circuit, then the circuit $P(T, \overline{x}, \phi_{k-1})$ is also constant-depth circuit and we can check for zeroness via known subexponential time PIT for constant-depth circuits. Unfortunately, we do not (yet) know if $\phi_{k-1}(T, \overline{x})$ is computable by small constant-depth circuits.

The main insight is that we can obtain a circuit for ϕ_{k-1} with enough structure so that $P(T, \overline{x}, \phi_{k-1}(T, \overline{x})) \stackrel{?}{=} 0$ can be tested via the same PIT algorithm. In particular, we analyze the standard Kabanets-Impagliazzo generator instantiated with a suitable hard polynomial for such structured circuits, and prove that it does preserve nonzeroness. The details can be found in Section 5.7.

5.2.2 Analyzing the KI generator on such circuits

The main structural lemma that drives our analysis is the following that shows that "low degree parts" of approximate roots can indeed be computed via not-too-large constant-depth circuits.

Lemma (Informal version of Lemma 5.6.3)**.** *Suppose $\phi_{k-1}(T, \overline{x})$ with $\deg_T \phi_{k-1} < k$ is an approximate z-root of $P(T, \overline{x}, z)$ as above. Then, for every integer $\ell \geqslant 0$, there is a circuit $C_\ell(T, \overline{x})$ of constant depth and "not-too-large" size that agrees with ϕ_{k-1} on all monomials of degree at most ℓ. That is,*

$$C_\ell(T, \overline{x}) = \phi_{k-1}(T, \overline{x}) \bmod \langle \overline{x} \rangle^\ell,$$

where "not-too-large" is $\mathrm{poly}(\mathrm{size}(P), \deg(P)) \cdot (\log k)^{\mathrm{poly}(\ell)}$.

While this is reminiscent of a lemma of [CKS19] (Lemma 5.4.7) that allows one to argue that low-degree components of a root have small constant-depth circuits, there is an important difference in our statement: $\phi_{k-1}(T, \overline{x})$ is a root modulo T^k, whereas we want a small constant-depth circuit to compute the root modulo $\langle \overline{x} \rangle^\ell$ i.e. a *different*

set of variables. This difference demands a little more care (see Section 5.6.2.1), although morally we still use a Taylor-expansion based approach along the lines of [CKS19].

With the above lemma, we would be able to show that the Kabanets-Impagliazzo generator will indeed preserve the nonzeroness of $P(T, \overline{x}, \phi_{k-1}(T, \overline{x}))$ when instantiated with a hard polynomial for constant-depth circuits.

Theorem (Informal version of Theorem 5.7.3)**.** *Let f be a polynomial that requires "large" constant-depth circuits. Then,*

$$P(T, \overline{x}, \phi_{k-1}(T, \overline{x})) \neq 0 \implies P(T, \overline{x}, \phi_{k-1}(T, \overline{x})) \circ \mathbf{KI}_f \neq 0$$

where $\mathbf{KI}_f : \mathbb{F}[\overline{x}] \to \mathbb{F}[\overline{w}]$ is the Kabanets-Impagliazzo generator instantiated with the polynomial f.

We now briefly outline why this is true. Let us define $R(T, \overline{x}) := P(T, \overline{x}, \phi_{k-1}(T, \overline{x}))$. To prove the above lemma, assume on the contrary that the generator $\mathbf{KI}_f$ did not preserve the non-zeroness of $R(T, \overline{x})$. By the standard hybrid argument (with additional substitutions), there are "easy-to-compute" polynomials f_i's such that

$$\begin{aligned} R(f_1, f_2, \ldots, f_{i-1}, x_i, f_{i+1}, \ldots, f_n, T) &\neq 0 \\ \text{but } R(f_1, f_2, \ldots, f_{i-1}, f, f_{i+1}, \ldots, f_n, T) &= 0. \end{aligned}$$

($f_{i+1}, \ldots, f_n$ are just field constants). Therefore, the polynomial $x_i - f$ divides the polynomial $R(f_1, \ldots, f_{i-1}, x_i, f_{i+1}, \ldots, f_n, T)$. At this point, we make use of the following result by Chou, Kumar and Solomon (stated informally).

Theorem (Chou-Kumar-Solomon (follows immediately from Lemma 5.4.7))**.** *Let $R(\overline{w}, y)$ be a non-zero polynomial with $R(\overline{w}, g(\overline{w})) = 0$ for some polynomial g of degree at most ℓ, with $\partial_y R(0, g(\overline{0})) \neq 0$.*
For all $i \leqslant \ell$, suppose there are small circuits $C_i(\overline{w}, y)$ such that

$$C_i(\overline{w}, y) = \partial_{y^i} R \bmod \langle \overline{w} \rangle^{\ell}.$$

If ℓ is "small", then g is a "small" composition of the circuits $C_1, \ldots, C_\ell$. In particular, if ℓ is small and each C_i is a small, constant-depth circuit, then g is also computable by a "small-ish", constant-depth circuit.

For our setting, let $\tilde{R}(\overline{w}, y, T) = R(f_1, f_2, \ldots, f_{i-1}, y, f_{i+1}, \ldots, f_n, T)$ which has $y - f(\overline{w})$ as a factor. Note that this is not a constant-depth circuit as $R(T, \overline{x}) = P(T, \overline{x}, \phi_{k-1}(T, \overline{x}))$ and ϕ_{k-1} is not known to have a constant-depth circuits. However, if $\deg(f)$ is small enough, then the above theorem of Chou, Kumar and Solomon asserts that we can construct a constant-depth circuit for $(y - f(\overline{w}))$ if we are able to build constant-depth circuits for $\mathrm{Hom}_{\leqslant \deg(f)}\left(\partial_{y^i} \tilde{R}(\overline{w}, y, T)\right)$. Fortunately, our main structural lemma states that low-degree homogeneous parts

of ϕ_{k-1} do have not-too-large constant-depth circuits. Putting this together, we are able to assert that $f(\overline{w})$ must also be computable by a not-too-large constant-depth circuit, thus contradicting the hardness of f.

5.2.3 Computing general factors

In Chapter 6, we see that these PIT instances, via the closure results of [Bha+25b], do have constant-depth circuits. Thus, any hitting-set generator preserves the nonzeroness of these instances.

For computing general factors, at a high level, the proof proceeds in three broad steps. We first show a way of characterizing irreducibility of polynomials by (exponentially many) divisibility tests involving the power series roots of these polynomials. We then show that these divisibility tests can be reduced to PIT instances for circuits that might not be constant-depth, but have some additional structure. And finally, in spite of being unable to show that these PIT instances are for constant-depth circuits, we manage to show that the Kabanets-Impagliazzo hitting-set generator [KI04] invoked with low degree hardness for constant-depth circuits from [LST25] preserves the non-zeroness of these PIT instances. This part of the proof again builds upon the techniques from [CKS19]. We now discuss these steps in a bit more detail and refer to Section 5.8 for the full proof.

The main technical insight of our proof is that the Kabanets-Impagliazzo generator invoked with low-degree hard polynomials for constant-depth circuits in fact preserves irreducibility of small constant-depth circuits and their factors. Let us consider the polynomial $P(\overline{0}, 0, z)$ over the algebraic closure $\overline{\mathbb{Q}}$, and let us assume this splits as $(z-\zeta_1)\cdots(z-\zeta_d)$. Let $\phi^{(1)}(T,\overline{x}),\ldots,\phi^{(d)}(T,\overline{x})$ be the approximate root lifted from ζ_i — i.e., it satisfies $P(T,\overline{x},\phi^{(i)}) = 0 \bmod T^k$ and $\phi^{(i)}(\overline{0},0) = \zeta_i$, where k is large enough. Then, each true factor $Q(T,\overline{x},z)$ of $P(T,\overline{x},z)$ corresponds to some subset S_Q of these approximate roots. That is, we must have

$$Q(T,\overline{x},z) = \prod_{i\in S_Q} (z-\phi^{(i)}(T,\overline{x})) \text{ trunc } T^k$$

where " trunc T^k" denotes the operation of discarding all monomials that have degree k or more in T. As a consequence, if $P(T,\overline{x},z)$ is indeed irreducible, then for every subset $\emptyset \neq S \subsetneq [d]$ we have $Q_S(T,\overline{x},z)$ does not divide $P(T,\overline{x},z)$ as a polynomial, where

$$Q_S(T,\overline{x},z) = \prod_{i\in S}(z-\phi^{(i)}(T,\overline{x})) \text{ trunc } T^k\,.$$

Turns out, the non-divisibility of such Q_S, P can be expressed as an appropriate PIT via standard reductions from divisibility testing to PIT [For15; AW24]. Once again, this is not a PIT of a constant-depth circuit due to the presence of the $\phi^{(i)}(T,\overline{x})$ sub-circuits which are not known to have small constant-depth circuits. Nevertheless, since we are able

to show that $\phi^{(i)}(T, \overline{x})$ has enough structure to allow us to build small constant-depth circuits for their "low-degree" components, we are able to argue that the Kabanets-Impagliazzo generator, when instantiated with a sufficiently hard low-degree polynomial, will preserve the non-zeroness of such PIT instances. Furthermore, our algorithms do not proceed by directly derandomizing these PIT instances since there are exponentially many of them to deal with. We only use this reduction in the analysis of our algorithms.

Overall, this yields a variable reduction (to n^{ϵ} variables, for any $\epsilon > 0$) that allows us to preserve the factorisation pattern of any constant-depth circuit. From this, recovering circuits (although not necessarily constant-depth) for the factors is relatively straightforward, given the known algorithms for polynomial factorisation.

Organization

The rest of the chapter is organized as follows. We begin by recalling some standard notations and preliminaries in Section 5.3, and some preliminaries related to polynomial factorisation in Section 5.4. We describe the details of our algorithms in Section 5.5, and discuss their analysis, assuming some technical results in Section 5.9. We discuss the proof of these technical results in detail in Section 5.6, Section 5.7 and Section 5.8.

For readers familiar with the general area of algebraic circuit complexity and some familiarity with polynomial factorisation algorithms, we recommend skipping the preliminaries and going to Section 5.5, and then referring to Section 5.3 and Section 5.4 as and when needed.

5.3 STANDARD PRELIMINARIES

5.3.1 Truncation

We start with the important, although non-standard definitions that are helpful in succinctly expressing our technical statements in the chapter.

Definition 5.3.1 (Polynomial truncation)**.** *For a polynomial* $Q(\overline{x}) \in R[\overline{x}]$ *and a positive integer* k, $Q \text{ trunc } \langle \overline{x} \rangle^k$ *denotes the unique polynomial* $\tilde{Q}$ *with* $\overline{x}$*-degree less than* k *such that* $\tilde{Q} \equiv Q \bmod \langle \overline{x} \rangle^k$*. We extend this notation to* $Q(T, \overline{x}) \in \mathbb{F}[\overline{x}][T]$ *(as well as power-series in* $\mathbb{F}[\overline{x}][\![T]\!]$*) and use* $Q \text{ trunc } T^k$ *to denote the truncation interpreting* $Q(T, \overline{x}) \in R[T]$ *(or* $R[\![T]\!]$*) for* $R = \mathbb{F}[\overline{x}]$*.* ◊

While it is common to slightly overload notation and use $\bmod \langle \overline{x} \rangle^k$ to also denote $\text{trunc } \langle \overline{x} \rangle^k$, we choose to separate these notations for reasons of clarity.

5.3.2 Consequences of interpolation

We recall some of the standard consequences of interpolation.

Corollary 5.3.2 (Standard consequences of interpolation)**.** *Let $\alpha_0, \ldots, \alpha_d$ be distinct elements in $\mathbb{F}$. Then,*

1. ***[Partial derivatives]*** *If $C(\overline{x}, y)$ has degree d in the variable y, then the i-th order partial derivative of C with respect to y can be expressed as an $\mathbb{F}[y]$-linear combination of $\{C(\overline{x}, \alpha_j) : j \in \{0, \ldots, d\}\}$. That is, there are polynomials $\mu_0(y), \ldots, \mu_d(y)$ (not depending on C) of degree at most d such that*

$$\partial_{y^i} C(\overline{x}, y) = \mu_0(y) \cdot C(\overline{x}, \alpha_0) + \cdots + \mu_d(\overline{y}) \cdot C(\overline{x}, \alpha_d).$$

2. ***[Homogeneous components]*** *Let $C(\overline{x})$ be a degree d polynomial. Then, for any subset $\overline{x}_S \subseteq \overline{x}$ and any $i \in [d]$, the degree i homogeneous part of C with respect to $\overline{x}_S$, denoted by $\mathrm{Hom}_{\overline{x}_S, i}(C)$, can be expressed as*

$$\mathrm{Hom}_{\overline{x}_S, i}(C) = \sum_{j=0}^{d} \beta_{i,j} \cdot C(\alpha_j \cdot \overline{x}_S, \overline{x}_{\overline{S}})$$

for some constants $\beta_{i,j} \in \mathbb{F}$ (not depending on C).

3. ***[Truncation]*** *Let $C(\overline{x})$ be a degree d polynomial and let $\overline{x}_S \subseteq \overline{x}$ be a subset of variables. Then, for any $\ell \in [d]$, the truncation $C(\overline{x})$ trunc $\langle \overline{x}_S \rangle^{\ell+1}$ can be expressed as*

$$C(\overline{x}) \text{ trunc } \langle \overline{x}_S \rangle^{\ell+1} = \sum_{j=0}^{d} \gamma_{\ell,j} \cdot C(\alpha_j \cdot \overline{x}_S, \overline{x}_{\overline{S}})$$

for some constants $\gamma_{\ell,j} \in \mathbb{F}$ (not depending on C).

In particular, if C is computable by a size s, depth Δ circuit, then all of the above operations yield a circuit of size $\mathrm{poly}(d, s)$ and depth $\Delta + O(1)$.

5.3.3 Strassen's division elimination

We recall a classical theorem of Strassen for division elimination in algebraic circuits.

Theorem 5.3.3 (Division elimination in algebraic circuits [Str73])**.** *Let C be an algebraic circuit of size s with division gates that computes a multivariate polynomial P of degree d over any sufficiently large field $\mathbb{F}$.*

Then, there is an algebraic circuit $\hat{C}$ of size at most $\mathrm{poly}(s, d)$ that computes the polynomial P and does not have any division gates.

As a matter of notation, algebraic circuits throughout this chapter do not have division gates. We explicitly mention if we have to deal with circuits with division gates at any point in our proof.

In our proofs, we rely on the following consequence of the above theorem. The proof easily follows from the standard proof of Theorem 5.3.3, for instance in [SY10].

Lemma 5.3.4 (Algorithmic division elimination)**.** *Let $\mathbb{F}$ be any field.*

There is a deterministic algorithm that takes as input algebraic circuits of size at most s computing n variate polynomials A, B over $\mathbb{F}$, a point $\overline{u} \in \mathbb{F}^n$ and a degree parameter d such that (a) B divides A, (b) the quotient A/B has degree at most d, and (c) $B(\overline{u}) \neq 0$, and outputs a (division free) algebraic circuit for the quotient A/B. Moreover, the algorithm runs in time $\mathrm{poly}(s, d)$.

5.3.4 Discriminant

A special case of the resultant that is very natural to study is the resultant of a polynomial $P(z)$ and its derivative $\frac{dP}{dz}$. This resultant is referred to as the *discriminant* of P, and unless the derivative vanishes for trivial reasons (for instance over fields of small characteristic), the discriminant captures the squarefreeness of P. More precisely, we have the following theorem.

Theorem 5.3.5 (Discriminant and squarefreeness [DSS22, Lemma 12])**.** *Let $\mathbb{F}$ be any field of characteristic zero, and let $P(z)$ be a univariate over $\mathbb{F}$ of degree at least one. Then, $P(z)$ is squarefree if and only if its discriminant* $\mathrm{Res}_z(P, \frac{\partial P}{\partial z})$ *is non-zero.*

A beautiful result of Andrews and Wigderson shows that the resultant (and hence, the discriminant) can be computed by a constant-depth circuit.

Theorem 5.3.6 (Computing resultants via constant-depth circuits [AW24, Theorem 6.1])**.** *Let $\mathbb{F}$ be a field of characteristic zero. For a fixed $\Delta \in \mathbb{N}$, there is a family of depth-Δ circuits $\{C_n\}_{n \in \mathbb{N}}$ with size $\leqslant \mathrm{poly}(n)$ such that the following is true. If $f, g \in \mathbb{F}[z]$ are degree-n univariate polynomials given by their coefficients, then C_n takes the coefficients of f and g as input, and computes the resultant* $\mathrm{Res}_z(f, g)$.

5.3.5 The Kabanets-Impagliazzo generator

The Kabanets-Impagliazzo hitting-set generator [KI04] for algebraic circuits is an adaptation of the Nisan-Wigderson generator in classical complexity to the algebraic setting. It allows us to obtain non-trivial derandomization for PIT for algebraic circuits from sufficiently hard explicit polynomial families. More precisely, Kabanets & Impagliazzo showed in [KI04] that given an explicit polynomial family that requires exponential size algebraic circuits, there is a deterministic algorithm for PIT for algebraic circuits (with size and degree polynomially bounded in the number of variables) that runs in quasipolynomial time.

This generator and the details of its analysis play an important role in the proofs in this chapter. We start by recalling the notion of combinatorial designs, and then define the generator.

Definition 5.3.7 (Combinatorial designs). *Let $n, \sigma, \mu, \rho \in \mathbb{N}$. A family of subsets $\mathcal{S} = (S_1, \ldots, S_n)$ is a (n, σ, μ, ρ) design if:*

- $\forall i \in [n] : S_i \subseteq [\mu]$
- $\forall i \in [n] : |S_i| = \sigma$
- *For any $i, j \in [n]$ s.t. $i \neq j : |S_i \cap S_j| < \rho$*

$\Diamond$

The following theorem gives an explicit construction of such designs.

Lemma 5.3.8 (Explicit construction of designs [NW94]). *For any positive integers n, σ with $n < 2^\sigma$, there exists an explicit (n, σ, μ, ρ)-design with $\mu = \dfrac{\sigma^2}{\log n}$ and $\rho = \log n$.*

Moreover, each subset inside the design can be computed in $\mathrm{poly}(n, 2^\mu)$ time deterministically.

Definition 5.3.9 (KI-generator $\mathbf{KI}_{g,\mathcal{S}}$ [KI04]). *Let $\mathbb{F}$ be a field, n, σ be positive integers with $n < 2^\sigma$ and $g(\overline{x})$ be a σ-variate polynomial. Let $\mathcal{S} = (S_1, \ldots, S_n)$ be an explicit (n, σ, μ, ρ)-design from Lemma 5.3.8. The Kabanets-Impagliazzo generator given by g and $\mathcal{S}$ is a polynomial map $\mathbf{KI}_{g,\mathcal{S}} : \mathbb{F}^\mu \to \mathbb{F}^n$ defined by $\overline{w} \mapsto (g_1(\overline{w}), \ldots, g_n(\overline{w}))$, where for each $i \in [n]$, $g_i(\overline{w}) := g(\overline{w}_{S_i})$ for $\overline{w}_{S_i} := (w_j : j \in S_i)$ is an ordered list with $w_a < w_b$ iff $a < b$. Thus, $\mathbf{KI}_{g,\mathcal{S}}$ also defines a homomorphism from $\mathbb{F}[x_1, \ldots, x_n]$ to $\mathbb{F}[w_1, \ldots, w_\mu]$ that maps each x_i to $g_i(\overline{w})$.* $\Diamond$

5.3.6 Lower bounds for constant-depth circuits

We now recall the results of [LST25] that prove superpolynomial lower bounds for constant-depth circuits.

Theorem 5.3.10 (Lower bounds for constant-depth circuits [LST25, Corollary 4]). *Suppose $n, d \in \mathbb{N}$ with $d \leqslant \log n/100$ and $\mathbb{F}$ is a field with $\mathrm{char}(\mathbb{F}) = 0$ or greater than d. Then, for any product-depth $\Delta \in \mathbb{N}$, there exists an explicit n-variate polynomial $P(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d such that any algebraic circuit of product-depth at most Δ must have size at least $n^{d^{\exp(-O(\Delta))}}$.*

5.4 PRELIMINARIES FOR POLYNOMIAL FACTORISATION

5.4.1 Regularized polynomials

Definition 5.4.1 (T-regularized polynomials and non-degenerate, truncated, approximate z-roots of order k)**.** *Let* $P(T, \overline{x}, z)$ *be a polynomial in* $\mathbb{F}[T, \overline{x}, z]$ *that is monic in the variable* z*, and* $\Phi(T, \overline{x}) \in \mathbb{F}[\overline{x}][\![T]\!]$ *be a power series.*

- $\Phi(T, \overline{x})$ *is said to be an* approximate z-root of order k with respect to T *if*

$$P(T, \overline{x}, \Phi(T, \overline{x})) \equiv 0 \bmod T^k.$$

  Moreover, an approximate z*-root* Φ *of order* k *is* truncated *if* $\deg_T \Phi < k$*. Throughout the chapter, approximate roots will be defined modulo* T^k*. For the sake of brevity, we sometimes refer to an approximate* z*-root of order* k *with respect to* T *as just an approximate* z*-root of order* k*.*

- $P(T, \overline{x}, z)$ *is* T-regularized with respect to z *if* $P(0, \overline{x}, z) \in \mathbb{F}[z]$*, that is, every monomial that depends on* $\overline{x}$ *is divisible by* T*.*

- $\Phi(T, \overline{x})$ *is a* non-degenerate *approximate* z*-root of* $P(T, \overline{x}, z)$ *if*

$$(\partial_z P)(0, \overline{0}, \alpha) \neq 0$$

  where $\alpha = \Phi(0, \overline{0}) \in \mathbb{F}$*; that is, the constant term of* Φ *is not a repeated root of* $P(0, \overline{0}, z)$*.* ◊

The following simple observation follows immediately from the above definitions.

Observation 5.4.2. *Let* $P(T, \overline{x}, z)$ *be a polynomial in* $\mathbb{F}[T, \overline{x}, z]$ *that is monic in the variable* z*,* T*-regularized, and let* $\Phi(T, \overline{x}) \in \mathbb{F}[\overline{x}][\![T]\!]$ *be a power series. If* $\Phi(T, \overline{x})$ *is an approximate* z*-root of* $P(T, \overline{x}, z)$*, then* $\Phi(0, \overline{x})$ *– a root of the univariate* $P(0, \overline{x}, z) \in \mathbb{F}[z]$ *– is a scalar in* $\mathbb{F}$*, implying that* $\Phi(0, \overline{x}) = \Phi(0, \overline{0})$*. Moreover,* $(\partial_z P)(0, \overline{0}, z) = (\partial_z P)(0, \overline{x}, z)$*.*

5.4.2 Newton iteration

We now recall various flavors of Newton iteration that we use in our proofs.

Lemma 5.4.3 (Newton iteration with linear convergence [CKS19, Lemma 5.1])**.** *Let* $R = \mathbb{F}[\overline{x}]$ *be a polynomial ring, and let* $H(\overline{x}, z) \in R[z]$*.*

Suppose $\phi \in \mathbb{F}[\![\overline{x}]\!]$ *is a power-series such that* $H(\overline{x}, \phi) = 0 \bmod \langle \overline{x} \rangle^m$ *and* $\partial_z H(\overline{0}, \phi(\overline{0})) \neq 0$. *Then,*

$$\phi' := \phi - \frac{H(\overline{x}, \phi)}{\partial_z H(\overline{0}, \phi(\overline{0}))}$$

satisfies $H(\overline{x}, \phi') = 0 \bmod \langle \overline{x} \rangle^{m+1}$ *and* $\phi' = \phi \bmod \langle \overline{x} \rangle^m$. *Furthermore, such an extension* ϕ' *of* ϕ *is unique in the sense that any* ϕ'' *that satisfies* $H(\overline{x}, \phi'') = 0 \bmod \langle \overline{x} \rangle^{m+1}$ *and* $\phi'' = \phi \bmod \langle \overline{x} \rangle^m$ *must satisfy*

$$\phi' = \phi'' \bmod \langle \overline{x} \rangle^{m+1}.$$

Corollary 5.4.4 ([CKS19, Corollary 5.5], [Kum+24, Lemma 3.1])**.** *Let* $R = \mathbb{F}[\overline{x}]$ *be a polynomial ring, and let* $H(\overline{x}, z) \in R[z]$ *be a polynomial of degree* D *and a circuit of size* s. *Suppose* $u \in \mathbb{F}$ *such that:*

$$H(\overline{0}, u) = 0$$
$$\frac{\partial H}{\partial z}(\overline{0}, u) \neq 0$$

Then for every $k \in \mathbb{N}$, *there is a unique truncated, non-degenerate, approximate z-root* $\Phi_k(\overline{x})$ *of order* k *with respect to* $\overline{x}$ *for* $H(\overline{x}, z)$, *satisfying* $\Phi_k(\overline{0}) = u$. *Moreover, there is a deterministic algorithm that runs in time* $\mathrm{poly}(s, D, k)$ *and outputs a circuit of size* $\mathrm{poly}(s, D, k)$ *for* $\Phi_k(\overline{x})$.

Lemma 5.4.5 (Newton iteration with quadratic convergence [GG13, Lemma 9.21, Lemma 9.27])**.** *Let* $R = \mathbb{F}[\overline{x}]$ *be a polynomial ring, and let* $H(\overline{x}, z) \in R[z]$.

Suppose $\phi \in \mathbb{F}[\![\overline{x}]\!]$ *is a power-series such that* $H(\overline{x}, \phi) = 0 \bmod \langle \overline{x} \rangle^m$ *and* $\partial_z H(\overline{0}, \phi(\overline{0})) \neq 0$. *Then, for any power series* $\sigma \in \mathbb{F}[\![\overline{x}]\!]$ *satisfying*

$$\sigma(\overline{x}) = \frac{1}{\partial_z H(\overline{x}, \phi)} \bmod \langle \overline{x} \rangle^m$$

we have that $\phi' := \phi - H(\overline{x}, \phi)\sigma$ *satisfies* $H(\overline{x}, \phi') = 0 \bmod \langle \overline{x} \rangle^{2m}$ *and* $\phi' = \phi \bmod \langle \overline{x} \rangle^m$. *Furthermore, such an extension* ϕ' *of* ϕ *is unique in the sense that any* ϕ'' *that satisfies* $H(\overline{x}, \phi'') = 0 \bmod \langle \overline{x} \rangle^{2m}$ *and* $\phi'' = \phi \bmod \langle \overline{x} \rangle^m$ *must satisfy*

$$\phi' = \phi'' \bmod \langle \overline{x} \rangle^{2m}.$$

Lemma 5.4.6 (Quadratic-convergence Newton Iteration without divisions)**.** *Let* $R = \mathbb{F}[\overline{x}]$ *be a polynomial ring, and let* $H(\overline{x}, z) \in R[z]$. *Suppose there exists* $\alpha \in \mathbb{F}$ *such that* $H(\overline{0}, \alpha) = 0$ *and* $\partial_z H(\overline{0}, \alpha) = \beta \neq 0$. *For each* $i \geqslant 0$, *define polynomials* $\phi_i, \sigma_i \in \mathbb{F}[\overline{x}]$ *as follows:*

$$\phi_0 := \alpha, \qquad \sigma_0 := (1/\beta),$$
$$\text{For } i \geqslant 0, \quad \phi_{i+1} := \phi_i - H(\overline{x}, \phi_i) \cdot \sigma_i, \quad \sigma_{i+1} := 2\sigma_i - \sigma_i^2 \cdot \partial_z H(\overline{x}, \phi_{i+1}).$$

Then $H(\overline{x}, \phi_i) = 0 \bmod \langle \overline{x} \rangle^{2^i}$, $\phi_{i+1} = \phi_i \bmod \langle \overline{x} \rangle^{2^i}$, *and* $\sigma_i \cdot \partial_z H(\overline{x}, \varphi_i) = 1 \bmod \langle \overline{x} \rangle^{2^i}$.

Proof. We prove this by induction on i. The base case $i = 0$ follows by definition.
Now suppose that for some $i \geqslant 0$, $H(\overline{x}, \phi_i) = 0 \bmod \langle \overline{x} \rangle^{2^i}$ and $\sigma_i \cdot \partial_z H(\overline{x}, \varphi_i) = 1 \bmod \langle \overline{x} \rangle^{2^i}$.
By Lemma 5.4.5, it follows that $\varphi_{i+1} := \phi_i - H(\overline{x}, \phi_i) \cdot \sigma_i$ satisfies $H(\overline{x}, \varphi_{i+1}) = 0 \bmod \langle \overline{x} \rangle^{2^{i+1}}$ and $\varphi_{i+1} = \varphi_i \bmod \langle \overline{x} \rangle^{2^i}$.

$$\begin{aligned}
\sigma_{i+1} \cdot \partial_z H(\overline{x}, \phi_{i+1}) - 1 &= (2\sigma_i - \sigma_i^2 \cdot \partial_z H(\overline{x}, \phi_{i+1})) \cdot \partial_z H(\overline{x}, \phi_{i+1}) - 1 \\
&= (\sigma_i \cdot \partial_z H(\overline{x}, \phi_{i+1}) - 1) - \left((\sigma_i \cdot \partial_z H(\overline{x}, \phi_{i+1}))^2 - \sigma_i \cdot \partial_z H(\overline{x}, \phi_{i+1})\right) \\
&= -(\sigma_i \cdot \partial_z H(\overline{x}, \phi_{i+1}) - 1)^2 \\
&= 0 \bmod \langle \overline{x} \rangle^{2^{i+1}}
\end{aligned}$$

where the last equality follows because $\varphi_{i+1} = \varphi_i \bmod \langle \overline{x} \rangle^{2^i}$ and $\sigma_i \cdot \partial_z H(\overline{x}, \phi_i) - 1 = 0 \bmod \langle \overline{x} \rangle^{2^i}$. □

5.4.3 A lemma of Chou, Kumar & Solomon

The following technical lemma of Chou, Kumar & Solomon [CKS19] is used in the analysis of Kabanets-Impagliazzo generator for constant-depth circuits. We also use technical variants of this lemma in the heart of our proofs (for instance, see Claim 5.6.4).

Lemma 5.4.7 (Lemma 5.2 and Lemma 5.3 in [CKS19]). *Let* $P \in \mathbb{F}[\overline{x}, y]$ *and let* $R(\overline{x})$ *be polynomials such that* R *is of degree at most* d, $P(\overline{x}, R(\overline{x})) \equiv 0$ *and* $\frac{\partial P}{\partial y}(\overline{0}, R(\overline{0}))$ *is non-zero.*

Then, there exists a $(d+1)$*-variate polynomial* $Q(\overline{z})$ *of degree at most* d *such that*

$$R(\overline{x}) \equiv Q(h_0(\overline{x}), h_1(\overline{x}), \ldots, h_d(\overline{x})) \mod \langle \overline{x} \rangle^{d+1},$$

where for every $i \in \{0, 1, \ldots, d\}$, $h_i(\overline{x})$ *is defined as*

$$h_i(\overline{x}) := \frac{\partial^j P}{\partial y^j}(\overline{x}, R(\overline{0})) - \frac{\partial^j P}{\partial y^j}(\overline{0}, R(\overline{0})) \operatorname{trunc} \langle \overline{x} \rangle^{d+1}.$$

For our proofs, we end up invoking Lemma 5.4.7 in settings where the polynomial P also depends on an additional variable T (whereas R does not). In this case, the quantity $\frac{\partial P}{\partial y}(T, \overline{0}, R(\overline{0}))$ potentially depends on the variable T. Since our root R does not depend on the variable T, we can set T to some field constant without disturbing the starting conditions for Newton iteration, and thus we can perform Newton iteration as usual.

Lemma 5.4.8. *Let* $P \in \mathbb{F}[T, \overline{x}, y]$ *and let* $R(\overline{x})$ *be polynomials such that* R *is of degree at most* d, $P(T, \overline{x}, R(\overline{x})) \equiv 0$ *and* $\frac{\partial P}{\partial y}(T, \overline{0}, R(\overline{0}))$ *is non-zero.*

Then, there exists a $\kappa \in \mathbb{F}$ *and a* $(d+1)$*-variate polynomial* $Q(\bar{z}) \in \mathbb{F}[\bar{z}]$ *of degree at most* d *such that*

$$R(\bar{x}) \equiv Q(h_0(\kappa, \bar{x}), h_1(\kappa, \bar{x}), \ldots, h_d(\kappa, \bar{x})) \mod \langle \bar{x} \rangle^{d+1},$$

where for every $i \in \{0, 1, \ldots, d\}$, $h_i(\bar{x})$ *is defined as*

$$h_i(T, \bar{x}) := \frac{\partial P}{\partial y^i}(T, \bar{x}, R(\bar{0})) - \frac{\partial P}{\partial y^i}(T, \bar{0}, R(\bar{0})) \text{ trunc } \langle \bar{x} \rangle^{d+1}.$$

Proof. Since $\frac{\partial P}{\partial y}(T, \bar{0}, R(\bar{0})) = \delta(T) \in \mathbb{F}[T]$ for some $\delta(T) \not\equiv 0$, there exists a κ such that $\delta(\kappa) = \delta_0 \neq 0$. Thus, the polynomial $\tilde{P}(\bar{x}, y) := P(\kappa, \bar{x}, y)$ satisfies $\tilde{P}(\bar{x}, R(\bar{x})) \equiv 0$ and $\frac{\partial \tilde{P}}{\partial y}(\bar{0}, R(\bar{0})) = \delta_0 \neq 0$, for some $\delta_0 \in \mathbb{F}$. Applying Lemma 5.4.7 on $\tilde{P}$ and $R(\bar{x})$ tells us that there exists a $(d+1)$-variate polynomial $Q(\bar{z})$ of degree at most d such that

$$R(\bar{x}) \equiv Q(h_0(\bar{x}), h_1(\bar{x}), \ldots, h_d(\bar{x})) \mod \langle \bar{x} \rangle^{d+1},$$

where for every $i \in \{0, 1, \ldots, d\}$, $h_i(\bar{x})$ is defined as

$$h_i(\bar{x}) := \frac{\partial^j \tilde{P}}{\partial y^j}(\bar{x}, R(\bar{0})) - \frac{\partial^j \tilde{P}}{\partial y^j}(\bar{0}, R(\bar{0})) \text{ trunc } \langle \bar{x} \rangle^{d+1}.$$

Since $\tilde{P}(\bar{x}, y) := P(\kappa, \bar{x}, y)$, the required statement follows. □

For most of this chapter, the only operation we are allowed on the T-variable is scaling by a field element since we care about roots $\bmod T^k$, and we want to preserve the T-degree of monomials during any such operations/substitutions. But the above lemma will be invoked at a point when we are concerned about roots $\mod \langle \bar{x} \rangle^d$, not $\bmod T^k$, which is why it will be okay to replace T by a field element κ.

5.4.4 Deterministic factorisation

For our proofs, we rely on a deterministic algorithm for factoring n variate degree d polynomials that run in time $d^{O(n)}$. This is implicit in many known algorithms for polynomial factorisation, for instance, in the results of Kopparty, Saraf, Shpilka [KSS15]. Such a statement can also be inferred from our proofs in this chapter. We recall a formal statement of this nature from a work of Lecerf below.

Theorem 5.4.9 ([Lec07, Proposition 4]). *Suppose* $P(x_1, \ldots, x_n, y) \in \mathbb{Q}[\bar{x}, y]$ *be a polynomial that is monic in* y *with total degree* d*. Further, suppose that* P *is squarefree and* $P(\bar{0}, y)$ *is squarefree. Then, there is a deterministic algorithm that takes* P *as input in the dense representation and outputs each of its irreducible factors in time* $\leqslant O(N^2)$*, where* $N = \binom{n+d+1}{n}$.

Intuitively, the proof of the theorem follows from standard Hensel Lifting or Newton Iteration based algorithms for polynomial factorisation, and observing (as formally done in [KSS15]) that at every stage

of the algorithm, randomness is needed only for polynomial identity testing. Moreover, these PIT instances are all polynomials of degree $\mathrm{poly}(d)$ and on $O(n)$ variables, and hence by Lemma 2.3.10 can be solved deterministically in time $d^{O(n)}$.

5.5 ALGORITHM

5.5.1 Main theorems and high-level overview

First, we describe our main structural result, which states that the Kabanets-Impagliazzo hitting-set generator, when instantiated with a sufficiently hard low-degree polynomial and an appropriate combinatorial design, preserves the irreducibility of the factors of constant-depth circuits. The precise statement requires a few more conditions, and these conditions are without loss of generality.

Throughout the chapter, we will use $\mathcal{G} = \{g_m\}_{m\in\mathbb{N}}$ to denote a family of explicit polynomials such that for every $m \in \mathbb{N}$, $g_m \in \mathbb{F}[x_1, \ldots, x_m]$, $d_m := \deg(g_m) \leqslant O(\log\log(m))$. Further, $\mathcal{G}$ has the property that for any depth $\Delta \in \mathbb{N}$, if $\mathcal{C} = \{C_m\}_{m\in\mathbb{N}}$ is a family of depth-Δ circuits computing $\mathcal{G}$, then $\mathcal{C}$ requires size $m^{d_m^{\exp(-O(\Delta))}}$, which is $m^{\omega(1)}$. Theorem 5.3.10 gives us such a family of explicit low-degree polynomials that are hard for constant-depth circuits.

Theorem 5.5.1 (Irreducibility-preserving variable reduction)**.** *Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$. Let $n \in \mathbb{N}$ and $\overline{x} := (x_1, \ldots, x_n)$. Let $P(T, \overline{x}, z)$ be a nonzero polynomial with the following properties.*

- *$P(T, \overline{x}, z)$ is computable by a size $s \leqslant \mathrm{poly}(n)$ and depth Δ circuit.*
- *$P(T, \overline{x}, z)$ is monic in z and T-regularized, with $\deg(P) = D \leqslant \mathrm{poly}(n)$.*
- *$P(T, \overline{x}, z)$ and $P(0, \overline{x}, z) = P(0, \overline{0}, z)$ are squarefree.*

Let $\sigma = O(n^{\varepsilon}), \mu = O(\frac{n^{2\varepsilon}}{\log(n)}), \rho = O(\log(n))$, and let $\mathcal{S}$ be an (n, σ, μ, ρ)-design. Let $\mathbf{KI}_{g_\sigma,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ be the polynomial map in Definition 5.3.9 defined using the design $\mathcal{S}$ and the polynomial g_σ from the family of hard polynomials $\mathcal{G}$. Then, for an absolute constant[3] $\beta_{\Delta,\varepsilon} \in \mathbb{N}$ and $n \geqslant \beta_{\Delta,\varepsilon}$, the following is true.

A candidate factor[4] $F(T, \overline{x}, z)$ is an irreducible factor of $P(T, \overline{x}, z)$ if and only if $F(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$ is an irreducible factor of $P(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$.

Informally, a candidate factor (defined in Definition 5.8.2) is a polynomial built using an arbitrary subset of the power-series roots of $P(T, \overline{x}, z)$. We show in Section 5.8 that these are the only polynomials

3 The precise value of $\beta_{\Delta,\varepsilon}$ depends on the exact hardness of the polynomial in $\mathcal{G}$ and the upper bounds obtained in our proofs.

4 See Definition 5.8.2

we need to care about while trying to prove that a generator preserves irreducibility.

A few remarks regarding the choice of parameters in Theorem 5.5.1:

- The family $\mathcal{G}$ has degree $d_m \leqslant O(\log\log(m))$ so that $(\log(m))^{\mathrm{poly}(d_m)}$ is $\mathrm{poly}(m)$. We can work with any $d_m \leqslant O((\log m)^{\alpha})$ for some small enough α depending on the exponent in $\mathrm{poly}(d_m)$, but $\log\log(m)$ works in every case and makes it simpler to state the theorems.

- The design $\mathcal{S}$ has $\sigma = O(n^{\varepsilon})$ because $\mathcal{G}$ is guaranteed to be superpolynomially hard for constant-depth circuits. Stronger hardness guarantees can be used with smaller σ to get algorithms with better time complexity, as is usually the case in hardness-vs-randomness results.

We now describe our main algorithmic result. Informally, the result states that for every choice of depth $\Delta \in \mathbb{N}$, there is a *deterministic* algorithm $\mathcal{A}_{\Delta}$ such that $\mathcal{A}_{\Delta}$ takes a depth-Δ circuit for a polynomial P as input, and outputs small (but unbounded depth) circuits along with multiplicity information for each irreducible factor of P. Moreover, $\mathcal{A}_{\Delta}$ runs in time subexponential in the input size.

Theorem 5.5.2 (Deterministic subexponential time algorithm for factorisation of constant-depth circuits)**.** *Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$. There exists an algorithm $\mathcal{A}_{\Delta,\varepsilon}$ which, for all sufficiently large n,*

- *takes as input a polynomial $P(\overline{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$ of degree $D \leqslant \mathrm{poly}(n)$ with a depth-Δ, size $s \leqslant \mathrm{poly}(n)$ circuit;*

- *outputs $\mathrm{poly}(s, D)$-sized circuits for each irreducible factor of P, along with the multiplicity of each such factor; and*

- *runs in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$.*

5.5.1.1 High-level overview of the algorithm

1. Suppose $P(\overline{x})$ is the polynomial that we would like to factor. We first use the algorithm by Andrews and Wigderson (Theorem 4.2.2) to compute the squarefree decomposition of P. Now, we deal with each squarefree part separately.

2. For a specific squarefree part $P_r(\overline{x})$, we perform some of the standard transformations ($x_i \mapsto T \cdot x_i + a_i \cdot z + b_i$) so that the polynomial $P_r(T, \overline{x}, z)$ is monic in z variable, T-regularized, and $P_r(0, \overline{x}, z) = P_r(0, \overline{0}, z)$ is squarefree.

3. We apply $\mathbf{KI}_{g,\mathcal{S}}(\overline{w})$ on $P_r(T, \overline{x}, z)$ (instantiated with an appropriately chosen low-degree hard polynomial g and a design $\mathcal{S}$). This maintains the irreducibility of each factor (Theorem 5.5.1). At this point, we are working with an n^{ε}-variate polynomial for some

$\varepsilon \in (0,1)$. Thus, we can use a brute-force / dense-representation factorisation algorithm to factorise $P_r(T, \mathbf{KI}_{g,s}(\overline{w}), z)$ in subexponential time (Theorem 5.4.9).

4. Since $P_r(0, \overline{x}, z) = P_r(0, \mathbf{KI}_{g,s}(\overline{w}), z) = P_r(0, \overline{0}, z)$, taking an irreducible factor of $P_r(T, \mathbf{KI}_{g,s}(\overline{w}), z)$ and setting T to 0 precisely tells us $G(0, \overline{x}, z)$ for each irreducible factor $G(T, \overline{x}, z)$ of $P_r(T, \overline{x}, z)$. Thus, we now have $G(0, \overline{x}, z) = G(0, \overline{0}, z)$ for each G that is an irreducible factor of P, and we can deal with each G separately.[5]

5. Suppose $G(T, \overline{x}, z)$ is an irreducible factor, and we have access to the univariate $G(0, \overline{x}, z) = G(0, \overline{0}, z)$. We factorise $G(0, \overline{0}, z)$ using Theorem 2.3.27. $G(0, \overline{0}, z)$ could have a linear factor of the form $(z - \alpha)$; in this case, we use Newton iteration (Corollary 5.4.4) to lift the root α to a truncated, approximate z-root $\varphi(T, \overline{x})$ of $P(T, \overline{x}, z)$ of sufficiently high accuracy. But all the factors of $G(0, \overline{0}, z)$ might be non-linear. In this situation, we artificially *add a root* u of $G(0, \overline{0}, z)$ to the field; more precisely, if $H(z)$ is an arbitrary irreducible factor of $G(0, \overline{0}, z)$, we work over the field $\frac{Q[u]}{H(u)}$ so that u is now a root of $G(0, \overline{0}, z)$. We can efficiently simulate arithmetic in this field. Thus, we can lift the root u to a truncated, approximate z-root $\varphi(T, \overline{x})$ of $P(T, \overline{x}, z)$ of sufficiently high accuracy.

6. (Lemma 5.9.1) Given a truncated, approximate z-root φ of $P(T, \overline{x}, z)$, we set up a linear system whose solution will give us a minimal polynomial of the root φ. Since we know $G(0, \overline{x}, z)$, we know the z-degree of G; this is essentially the information we need to set up the linear system in a way that ensures that the solution is the same as the irreducible factor G.

7. (Theorem 5.9.2) The solution to the linear system can be represented as a small circuit using Cramer's rule, but this involves a division by a determinant. We shall now use Strassen's division elimination to represent the solution as an arithmetic circuit without division gates; this requires a point where the denominator evaluates to a nonzero value. To this end, we again compose the denominator with $\mathbf{KI}_{g,s}$ to get an n^{ε}-variate polynomial; since this denominator essentially captures the uniqueness of the minimal polynomial, we can prove that $\mathbf{KI}_{g,s}$, by maintaining irreducibility of factors, also maintains the non-zeroness of

5 At this point, with the right univariate projections $G(0, \overline{0}, z)$ for each irreducible factor $G(T, \overline{x}, z)$, one way to get the irreducible factors is to perform *Hensel lifting* (for instance, see [Sud98, Lecture 7] or [ST20]). While Hensel lifting usually ends with a *reconstruction step* that involves solving a linear system, the clean-up for us would just be a truncation since Hensel lifting guarantees that if we start off with the right univariate projection, we will retrieve the right factor, modulo higher degree terms that might have accumulated in the process.

the denominator. We can now use a brute-force derandomization of the Polynomial Identity Lemma to find a point where the denominator evaluates to a nonzero value, and hence, carry out Strassen's division elimination. Thus, we have division-free circuits for each irreducible factor $G(T, \overline{x}, z)$ of $P(T, \overline{x}, z)$.

8. Finally, we can undo our initial transformations and output circuits for the irreducible factors of the input polynomial $P(\overline{x})$, along with their multiplicities.

5.5.2 The algorithm

Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$. The following algorithm is the "outermost" wrapper for our complete algorithm, and it can be thought of as the algorithm $\mathcal{A}_{\Delta,\varepsilon}$ in Theorem 5.5.2. The algorithm computes the squarefree decomposition of the input polynomial, and then uses Algorithm 6 on each squarefree part of the decomposition.

Algorithm 5: All factors of depth-Δ circuits, parameter $\varepsilon \in (0, 0.5)$

Input : A depth-Δ circuit $C_P(\overline{x})$ of size $s \in \mathbb{N}$, computing a degree-D polynomial $P(\overline{x}) = \prod_{i=1}^{m} G_i(\overline{x})^{e_i} \in \mathbb{Q}[\overline{x}]$, where $\overline{x} = (x_1, \ldots, x_n)$.

Output : A list $L = \{(C_{G_1}(\overline{x}), e_1), \ldots, (C_{G_m}(\overline{x}), e_m)\}$, such that each $C_{G_i}(\overline{x})$ is a circuit of size $\mathrm{poly}(s, D)$ and depth $\mathrm{poly}(D)$, which computes the irreducible factor $G_i(\overline{x})$, and e_i is the multiplicity of G_i in P.

1 Run Andrews-Wigderson (Theorem 4.2.2) on $C_P(\overline{x})$ to get circuits $C_{P_1}(\overline{x}), \ldots, C_{P_r}(\overline{x})$ of depth $\Delta' = \Delta + O(1)$, computing $P_1(\overline{x}), \ldots, P_r(\overline{x})$ where $r = \max_{i \in [m]} e_i$ and $P_i(\overline{x}) = \prod_{j \in S_i} G_j(\overline{x})$ for $S_i = \{j \in [m] : e_j = i\}$

2 Initialize $L \leftarrow \emptyset$

3 **forall** $i \in [r]$ **do**

4 Run Algorithm 6 for parameters Δ' and ε on $C_{P_i}(\overline{x})$ to get a list $L_i := \{C_{G_j}(\overline{x}) : j \in S_i\}$

5 For each $C_{G_j}(\overline{x}) \in L_i$, add $(C_{G_j}(\overline{x}), i)$ to L.

6 **return** L.

5.5.3 Algorithm for factors of squarefree polynomials

In this section, we will describe the algorithm for steps 3 and 4 of the overview in Section 5.5.1.1. Fix any $\Delta \in \mathbb{N}$.

Let $\mathcal{G} = \{g_m\}_{m \in \mathbb{N}}$ be a family of polynomials such that for every $m \in \mathbb{N}$, $g_m \in \mathbb{F}[x_1, \ldots, x_m]$, $d_m := \deg(g_m) \leqslant O(\log\log(m))$. Further, $\mathcal{G}$ has the property that for any depth $\Delta \in \mathbb{N}$, if $\mathcal{C} = \{C_m\}_{m \in \mathbb{N}}$

is a family of depth-Δ circuits computing $\mathcal{G}$, then $\mathcal{C}$ requires size $m^{d_m^{\exp(-O(\Delta))}}$, which is $m^{\omega(1)}$. Theorem 5.3.10 gives us such a family of explicit low-degree polynomials that are hard for constant-depth circuits.

Let $\varepsilon \in (0, 0.5)$. Let $\sigma = O(n^{\varepsilon}), \mu = O(\frac{n^{2\varepsilon}}{\log(n)}), \rho = O(\log(n))$, and let $\mathcal{S}$ be an (n, σ, μ, ρ)-design. Let $\mathbf{KI}_{g_\sigma,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ be the polynomial map in Definition 5.3.9 defined using the design $\mathcal{S}$ and the polynomial g_σ from the family of hard polynomials $\mathcal{G}$.

Algorithm 6: Factors of squarefree polynomial with depth-Δ circuits, parameter $\varepsilon \in (0, 0.5)$

Input : A depth-Δ size-s circuit C_P computing the squarefree degree-D polynomial $P(\overline{x}) = \prod_{j\in[m]} G_j(\overline{x})$, where the G_js are the irreducible factors of P.

Output : A list $L = \{C_{G_1}(\overline{x}), \ldots, C_{G_m}(\overline{x})\}$, such that each $C_{G_j}(\overline{x})$ is a circuit of size $\mathrm{poly}(s, D)$ and depth $\mathrm{poly}(D)$, which computes the irreducible factor $G_j(\overline{x})$.

1 Compute $\overline{a} \in \mathbb{Q}^n$ such that $\delta = \mathrm{Hom}_D[P](\overline{a}) \neq 0$ using Theorem 2.3.13. Let $\hat{P}(\overline{x}, z) := P(\overline{x} + (\overline{a} \cdot z))/\delta$ and for each $j \in [m]$, $\hat{G}_j(T, \overline{x}, z) := G_j(\overline{x} + (\overline{a} \cdot z))/\mathrm{Hom}_{\deg(G_j)}[G_j](\overline{a})$

`// Ensures that` $\hat{P}$ `is monic in` z

2 Compute $\overline{b} \in \mathbb{Q}^n$ such that $\mathrm{Disc}_z(\hat{P})(\overline{b}) \neq 0$ using Theorem 2.3.13. Let $\tilde{P}(T, \overline{x}, z) := \hat{P}((T \cdot \overline{x}) + \overline{b}, z)$ and for each $j \in [m]$, $\tilde{G}_j(T, \overline{x}, z) := \hat{G}_j((T \cdot \overline{x}) + \overline{b}, z)$

`// Ensures that` $\tilde{P}(T, \overline{x}, z)$ `is` T`-regularised, and` $P(0, \overline{0}, z)$ `is squarefree.`

3 Construct the map $\mathbf{KI}_{g_\sigma,\mathcal{S}}$ using Lemma 5.3.8 and $g_\sigma \in \mathcal{G}$ as given by Theorem 5.3.10.

4 Factorise $C_{\tilde{P}}(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ so that for each $j \in [m]$, $C_{\tilde{G}_j}(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ is a circuit that computes $\tilde{G}_j(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$, satisfying the property that $\tilde{G}_j(0, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z) = \tilde{G}_j(0, \overline{0}, z)$.

5 Run Algorithm 7 with parameters $\Delta + 2$ and ε on $C_{\tilde{P}}(T, \overline{x}, z)$ and $\{\tilde{G}_j(0, \overline{0}, z) : j \in [m]\}$ to get a list $L' = \{C_{\tilde{G}_1}(T, \overline{x}, z), \ldots, C_{\tilde{G}_m}(T, \overline{x}, z)\}$.

6 For each $C_{\tilde{G}_j}(T, \overline{x}, z) \in L'$, compute $C_{G_j}(\overline{x}) := C_{\tilde{G}_j}(1, \overline{x} - \overline{b}, 0)$.

7 Output $L = \{C_{G_1}(\overline{x}), \ldots, C_{G_m}(\overline{x})\}$

5.5.4 Algorithm for obtaining irreducible factors from the right univariate projections

In this section, we describe the algorithm for steps 5-7 of the overview in Section 5.5.1.1. Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$.

Algorithm 7: Factors of monic, T-regularized, squarefree polynomial with depth-Δ circuit, parameter $\varepsilon \in (0, 0.5)$

Input :

- A depth-Δ size-s circuit $C_{\tilde{P}}$ computing a squarefree degree-D polynomial $\tilde{P}(T, \overline{x}, z) = \prod_{j \in [m]} \tilde{G}_j(T, \overline{x}, z)$, where the $\tilde{G}_i$s are the irreducible factors of $\tilde{P}$. Further, $\tilde{P}(T, \overline{x}, z)$ is monic in z, T-regularized with respect to $\overline{x}$ and $C_{\tilde{P}}(0, \overline{x}, z) = C_{\tilde{P}}(0, \overline{0}, z)$ is squarefree.
- For each $j \in [m]$, the univariate polynomial $\tilde{G}_j(0, \overline{x}, z) = \tilde{G}_j(0, \overline{0}, z)$.

Output : A list $L = \{C_{\tilde{G}_1}(T, \overline{x}, z), \ldots, C_{\tilde{G}_m}(T, \overline{x}, z)\}$, such that each $C_{\tilde{G}_i}(T, \overline{x}, z)$ is a circuit of size $\mathrm{poly}(s, D)$ and depth $\mathrm{poly}(D)$, which computes the irreducible factor $\tilde{G}_i(T, \overline{x}, z)$ of $\tilde{P}(T, \overline{x}, z)$.

Prerequisites: $\mathbf{KI}_{g_{\sigma,\mathcal{G}}}$ constructed in Algorithm 6 using Lemma 5.3.8 and Theorem 5.3.10

Initialize $L \leftarrow \emptyset$
forall $j \in [m]$ **do**
  Factorise the degree-D_z polynomial $\tilde{G}_j(0, \overline{0}, z)$ (Theorem 2.3.27) to get $\tilde{G}_j(0, \overline{0}, z) = \prod_{l=1}^{r_j} H_l(z)$, where every $H_l(z)$ is distinct, monic and irreducible.
  Let $H(z)$ be an arbitrary irreducible factor of $\tilde{G}_j(0, \overline{0}, z)$. Let $\mathbb{K}$ be the field $\frac{\mathbb{Q}[u]}{H(u)}$.
  Use Newton iteration (Lemma 5.4.3) to compute a truncated, non-degenerate, approximate z-root $\varphi(T, \overline{x}) \in \mathbb{K}[T, \overline{x}]$ of $\tilde{G}_j(T, \overline{x}, z)$ of order $2D \cdot D_z + 1$ such that $\varphi(0, \overline{x}) = \varphi(0, \overline{0}) = u \in \mathbb{K}$.
  Use the algorithm from Theorem 5.9.2 (with $\mathbf{KI}_{g_{\sigma,\mathcal{G}}}$) on $\varphi(T, \overline{x})$ to compute a circuit $C_{\tilde{G}_j}$ for $\tilde{G}_j$.
  Add $C_{\tilde{G}_j}$ to L.
return L

5.6 TECHNICAL BUILDING BLOCKS

5.6.1 Properties of roots modulo T^k

The following lemma observes that certain homomorphisms defined by polynomial maps preserve approximate roots for a polynomial. This will be used to argue that an approximate root remains an approximate root (with some nice properties) even after we plug in the Kabanets-Impagliazzo generator.

Lemma 5.6.1 (Preserving root properties under homomorphisms)**.** *Let $k > 1$ be any natural number and let $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ and $R(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ be polynomials satisfying the following properties.*

- $P(T, \overline{x}, z)$ *is* T*-regularized and it is monic in* z.
- $R(T, \overline{x})$ *is a truncated, non-degenerate, approximate* z*-root of* $P(T, \overline{x}, y)$ *of order* k.

For a new tuple $\overline{w}$ *of* μ *variables distinct from* $T, \overline{x}, z$*, polynomials* $h_1, h_2, \ldots, h_n \in \mathbb{F}[\overline{w}]$ *and a non-zero field constant* $\gamma \in \mathbb{F}$*, let* $\Lambda : \mathbb{F}[T, \overline{x}, z] \to \mathbb{F}[T, \overline{w}, z]$ *be the ring homomorphism defined by* $T \mapsto \gamma T$*,* $x_i \mapsto h_i(\overline{w})$ *and* $z \mapsto z$*. Then, the following are true.*

- $\Lambda(R)(0, \overline{w}) = R(0, \overline{x}) \in \mathbb{F}$
- $\Lambda(P)(T, \overline{w}, z)$ *is* T*-regularized, and it is monic in* z.
- $\Lambda(R)(T, \overline{w})$ *is a truncated, non-degenerate, approximate* z*-root of* $\Lambda(P)(T, \overline{w}, y)$ *of order* k

Proof. The lemma follows essentially because Λ is a homomorphism.

1. By Observation 5.4.2, $R(0, \overline{x}) \in \mathbb{F}$. Since Λ is a homomorphism with the property that for any monomial $m \in \mathbb{F}[T, \overline{x}, z]$, $\deg_T(m) = \deg_T(\Lambda(m))$, it follows that $\Lambda(R)(0, \overline{w}) = R(0, \overline{x})$.
2. If a monomial $m \in \mathbb{F}[T, \overline{x}, z]$ is T-regularized, then so is $\Lambda(m)$. Since Λ is a homomorphism, this property extends to all polynomials. Similarly, since $\Lambda(z) = z$, $\Lambda(P)$ remains monic in z.
3. $P(T, \overline{x}, R(T, \overline{x})) \equiv 0 \mod T^k$, or equivalently, every monomial in $P(T, \overline{x}, R(T, \overline{x}))$ has T-degree at least k. As observed in the first point, Λ is a homomorphism with the property that for any monomial $m \in \mathbb{F}[T, \overline{x}, z]$, $\deg_T(m) = \deg_T(\Lambda(m))$. Thus, $\Lambda(P)(T, \overline{w}, \Lambda(R)(T, \overline{w})) \equiv 0 \mod T^k$.

□

The following lemma observes that truncated, non-degenerate, approximate roots of a polynomial are unique once the value of the root at $T = 0$ is decided; it follows almost immediately from the uniqueness of approximate roots computed via Newton iteration.

Lemma 5.6.2 (Uniqueness of truncated, non-degenerate, approximate roots)**.** *Let* $k \in \mathbb{N}$*. Suppose* $\Psi_1(T, \overline{x})$ *and* $\Psi_2(T, \overline{x})$ *are truncated, non-degenerate, approximate* z*-roots of order* k *with respect to* T *for a* T*-regularized polynomial* $P(T, \overline{x}, z)$*. If* $\Psi_1(0, \overline{x}) = \Psi_2(0, \overline{x}) = \alpha \in \mathbb{F}$*, then* $\Psi_1(T, \overline{x}) \equiv \Psi_2(T, \overline{x})$.

Proof. Since $\Psi_1(T, \overline{x})$ is a non-degenerate root of $P(T, \overline{x}, z)$ which is T-regularized, $\frac{\partial P}{\partial z}(0, \overline{x}, \Psi_1(0, \overline{x})) = \frac{\partial P}{\partial z}(0, \overline{0}, \alpha) = \beta \in \mathbb{F}$ is nonzero. Similarly, $\frac{\partial P}{\partial z}(0, \overline{x}, \Psi_2(0, \overline{x})) = \frac{\partial P}{\partial z}(0, \overline{0}, \alpha) = \beta$. Since α satisfies $P(T, \overline{x}, \alpha) \equiv 0 \bmod T$ and $\frac{\partial P}{\partial z}(0, \overline{x}, \alpha) = \beta \in \mathbb{F}$ for $\beta \neq 0$, Lemma 5.4.3 tells us that there is a unique truncated, approximate z-root $\Phi(T, \overline{x})$ of order k for $P(T, \overline{x}, z)$, satisfying $\Phi(0, \overline{x}) = \alpha$. Thus, $\Phi(T, \overline{x}) \equiv \Psi_1(T, \overline{x}) \equiv \Psi_2(T, \overline{x})$. □

5.6.2 Complexity of low degree homogeneous components of roots

The following is our main technical lemma where we argue that the low-degree homogeneous components of an approximate root for a constant-depth circuit can be computed by a small constant-depth circuit.

Lemma 5.6.3. *Let* $k > 1$ *be any natural number and let* $P(T, \overline{w}, z) \in \mathbb{F}[T, \overline{w}, z]$ *and* $R(T, \overline{w}) \in \mathbb{F}[T, \overline{w}]$ *be polynomials satisfying the following properties.*

- $P(T, \overline{w}, z)$ *is computable by a depth* Δ *circuit of size* s.
- $P(T, \overline{w}, z)$ *is* T*-regularized and monic in* z.
- $R(T, \overline{w})$ *is a truncated, non-degenerate, approximate* z*-root of* $P(T, \overline{x}, z)$ *of order* k *with respect to* T.

Then, for every $\ell \in \mathbb{N}$*, there is an algebraic circuit* $C_\ell \in \mathbb{F}[T, \overline{w}]$ *of depth at most* $(\Delta + O(1))$ *and size at most* $\left(\mathrm{poly}(s, \deg(P)) \cdot (\log k)^{\mathrm{poly}(\ell)}\right)$

$$C_\ell(T, \overline{w}) = R(T, \overline{w}) \text{ trunc } \langle \overline{w} \rangle^\ell .$$

Proof. As seen in Observation 5.4.2, $R(0, \overline{w})$ is a root of $P(0, \overline{0}, z)$, and hence must be a field element α (possibly from an extension of $\mathbb{F}$) and does not depend on $\overline{w}$. Similarly, we also get that $\frac{\partial P}{\partial z}(0, \overline{w}, R(0, \overline{w}))$ is a non-zero field element that we denote by β. Thus, $R(T, \overline{w})$ is the *unique* lift of the root α of $P(T, \overline{w}, z)$ modulo T to a root modulo the ideal T^k (Lemma 5.6.2). In particular, we can view R as an outcome of Newton Iteration (and then eventual truncation modulo T^k). We prove the lemma by induction. Let $2^m \in [k, 2k]$ be the smallest power of 2 greater than k. From Lemma 5.4.6, we get that the sequence of polynomials $\phi_0, \phi_1, \ldots, \phi_m, \psi_0, \psi_1, \ldots, \psi_m \in \mathbb{F}[T, \overline{w}]$ defined as

$$\phi_0 = \alpha, \qquad \psi_0 = (1/\beta),$$

$$\text{For } i \geqslant 0, \quad \phi_{i+1} = \phi_i - P(T, \overline{w}, \phi_i) \cdot \psi_i, \quad \psi_{i+1} = 2\psi_i - \psi_i^2 \cdot \frac{\partial P}{\partial z}(T, \overline{w}, \phi_{i+1}).$$

satisfy

$$R(T, \overline{w}) = \phi_m(T, \overline{w}) \text{ trunc } T^k .$$

We note that while we are dealing with polynomials in both T and $\overline{w}$ variables, the lifting is happening only with respect to T. In this sense, we are really viewing the polynomial $P(T, \overline{w}, z)$ as a polynomial in $\mathbb{F}[\overline{w}][T, z]$ for the purpose of this lifting. We now use the following claim, whose proof we defer to the end of this section, to complete the proof of the lemma.

Claim 5.6.4. *For every $i \geqslant 0$, there exists a set $\mathcal{G}_i$ of at most $\tau_i \leqslant 2(\ell+1)i$ polynomials $\{g_1, g_2, \ldots, g_{\tau_i}\}$ in $\mathbb{F}[T, \overline{w}]$ and two (τ_i+1)-variate polynomials $Q_i(T, u_1, \ldots, u_{\tau_i}), \hat{Q}_i(T, u_1, \ldots, u_{\tau_i}) \in \mathbb{F}[T, \overline{u}]$ such that*

$$\phi_i \equiv Q_i(T, g_1, \ldots, g_{\tau_i}) \mod \langle \overline{w} \rangle^{\ell},$$

and

$$\psi_i \equiv \hat{Q}_i(T, g_1, \ldots, g_{\tau_i}) \mod \langle \overline{w} \rangle^{\ell}.$$

Moreover, each polynomial g in $\mathcal{G}_i$ is of the form $\frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma)$ for some $j \leqslant (\ell+1)$ and $\gamma \in \mathbb{F}[T]$.

From the fact that $R(T, \overline{w}) = \phi_m(T, \overline{w}) \text{ trunc } T^k$, we get that

$$R(T, \overline{w}) \text{ trunc } \langle \overline{w} \rangle^{\ell} = \left(\phi_m(T, \overline{w}) \text{ trunc } T^k\right) \text{ trunc } \langle \overline{w} \rangle^{\ell}.$$

Now, since T and $\overline{w}$ are disjoint variables, we can exchange the order of the operations of truncating modulo T and truncating modulo $\langle \overline{w} \rangle^{\ell}$. So, we have that

$$R(T, \overline{w}) \text{ trunc } \langle \overline{w} \rangle^{\ell} = \left(\phi_m(T, \overline{w}) \text{ trunc } \langle \overline{w} \rangle^{\ell}\right) \text{ trunc } T^k.$$

From Claim 5.6.4, we get that

$$R(T, \overline{w}) \text{ trunc } \langle \overline{w} \rangle^{\ell} = \left(Q_m(T, g_1, g_2, \ldots, g_{\tau_m}) \text{ trunc } \langle \overline{w} \rangle^{\ell}\right) \text{ trunc } T^k.$$

We would like to show that the RHS of the above equation has a small constant-depth circuit. Since we eventually truncate to T-degree $k-1$, we can assume without loss of generality that the T-degree of Q_m and each g_i is at most $k-1$. In particular, if g_i equals $\frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma)$ for some $j \in \mathbb{N}$ and $\gamma \in \mathbb{F}[T]$, this γ can be assumed to be of T-degree at most $(k-1)$. Moreover, since $P(T, \overline{w}, z)$ is a polynomial with a depth-Δ circuit of size s, we get, from Corollary 5.3.2 that $\frac{\partial P}{\partial z^j}(T, \overline{w}, z)$ has a depth $(\Delta + O(1))$ circuit of size at most $s \cdot \text{poly}(\deg(P))$. From the bound on the degree of $\gamma \in \mathbb{F}[T]$, we get that each $g_i(T, \overline{w})$ has a depth $(\Delta + O(1))$ circuit of size at most $s \cdot \text{poly}(\deg(P))$.

Finally, we note that we can view $Q_m(T, g_1, g_2, \ldots, g_{\tau_m})$ as

$$\tilde{Q}_m(T, g_1(T, \overline{w}) - g_1(T, \overline{0}), g_2(T, \overline{w}) - g_2(T, \overline{0}), \ldots, g_{\tau_m}(T, \overline{w}) - g_{\tau_m}(T, \overline{0})),$$

for some polynomial $\tilde{Q}_m$. Since we are only interested in working with the above polynomial modulo $\langle \overline{w} \rangle^{\ell}$ and every monomial in each of the polynomials $g_j(T, \overline{w}) - g_j(T, \overline{0})$ has $\overline{w}$-degree at least one, we get that $\tilde{Q}_m(T, u_1, \ldots, u_{\tau_m})$ can be assumed to have degree at most ℓ in the $\overline{u}$ variables. Thus, $\tilde{Q}_m(T, u_1, \ldots, u_{\tau_m})$ can be computed by a depth-2 circuit of size at most $k \cdot \binom{\tau_m + \ell}{\ell} \leqslant k \cdot (\log k)^{\text{poly}(\ell)}$. Combining this circuit with the constant-depth circuits for $g_j(T, \overline{w})$ (and hence

$g_j(T, \overline{w}) - g_j(T, \bar{0})$) gives us a depth-$(\Delta + O(1))$ circuit $\tilde{C}$ of size at most $\mathrm{poly}(s, \deg(P)) \cdot (\log k)^{\mathrm{poly}(\ell)}$ satisfying

$$\tilde{C}(T, \overline{w}) \equiv Q_m(T, g_1, \ldots, g_{\tau_m}) \bmod T^k \bmod \langle \overline{w} \rangle^{\ell}.$$

The T-degree of this circuit is at most $\deg_T(\tilde{Q}_m) \cdot \ell \cdot k \leqslant \mathrm{poly}(k, \ell)$. The $\overline{w}$-degree of this circuit is at most $\ell \cdot \deg(P)$. Thus, we can apply the operations $\mathrm{trunc}\ \langle \overline{w} \rangle^{\ell+1}$ and $\mathrm{trunc}\ T^k$ by an application of Corollary 5.3.2 to get the circuit $C_\ell(T, \overline{w})$ in the conclusion of the lemma. □

We now discuss the proof of Claim 5.6.4.

Proof of Claim 5.6.4. We prove the claim via an induction on i. For $i = 0$, we already know that $\phi_0 = \alpha$ and $\psi_0 = \frac{1}{\beta}$ are polynomials in $\mathbb{F}[T]$ and hence the claim immediately holds. We now assume that the claim holds for ϕ_i and ψ_i i.e.

$$\phi_i \equiv Q_i(T, g_1, \ldots, g_{\tau_i}) \mod \langle \overline{w} \rangle^{\ell},$$

and

$$\psi_i \equiv \hat{Q}_i(T, g_1, \ldots, g_{\tau_i}) \mod \langle \overline{w} \rangle^{\ell}$$

and show that it must hold for ϕ_{i+1} and ψ_{i+1}.

From the inductive definitions of ϕ_{i+1} and ψ_{i+1}, we get that

$$\phi_{i+1} \equiv \phi_i - P(T, \overline{w}, \phi_i) \cdot \psi_i$$

and

$$\psi_{i+1} \equiv 2\psi_i - \psi_i^2 \cdot \frac{\partial P}{\partial z}(T, \overline{w}, \phi_{i+1})$$

If $\gamma_i := \phi_i(T, \bar{0})$, i.e. γ_i is the $\overline{w}$-free part of ϕ_i, we get via a Taylor expansion that

$$P(T, \overline{w}, \phi_i) = P(T, \overline{w}, \gamma_i + (\phi_i - \gamma_i)) = \sum_{j=0}^{\deg(P)} \frac{1}{j!} \cdot \frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_i) \cdot (\phi_i - \gamma_i)^j. \tag{5.6.5}$$

From the definition of γ_i, we have that every monomial in $(\phi_i - \gamma_i)$ has $\overline{w}$-degree at least one. Thus, for every $j \geqslant \ell$, we have that $(\phi_i - \gamma_i)^j \equiv 0$ modulo $\langle \overline{w} \rangle^{\ell}$. Thus, we have that

$$P(T, \overline{w}, \phi_i) \equiv \sum_{j=0}^{\ell} \frac{1}{j!} \cdot \frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_i) \cdot (\phi_i - \gamma_i)^j \mod \langle \overline{w} \rangle^{\ell}.$$

Similarly,

$$\frac{\partial P}{\partial z}(T, \overline{w}, \phi_{i+1}) \equiv \sum_{j=0}^{\ell} \frac{1}{j!} \cdot \frac{\partial P}{\partial z^{j+1}}(T, \overline{w}, \gamma_{i+1}) \cdot (\phi_{i+1} - \gamma_{i+1})^j \mod \langle \overline{w} \rangle^{\ell}.$$

Thus, ϕ_{i+1} can be written as a polynomial in $Q_i(T, g_1, \ldots, g_{\tau_i}), \hat{Q}_i(T, g_1, \ldots, g_{\tau_i})$ and the polynomials in the set $\{\frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_i) : j \in \{0, 1, \ldots, \ell\}\}$, where the coefficients of this polynomial are from the ring $\mathbb{F}[T]$ and all equalities hold modulo $\langle \overline{w} \rangle^\ell$. Similarly, ψ_{i+1} can be written as a polynomial in $Q_i(T, g_1, \ldots, g_{\tau_i}), \hat{Q}_i(T, g_1, \ldots, g_{\tau_i})$ and polynomials in the set $\{\frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_{i+1}) : j \in \{1, 2, \ldots, \ell+1\}\}$, where the coefficients of this polynomial are from the ring $\mathbb{F}[T]$ and all equalities hold modulo $\langle \overline{w} \rangle^\ell$. Thus, both ϕ_{i+1} and ψ_{i+1} can be written as polynomials in the *generating set*

$$\{g_1, \ldots, g_{\tau_i}\} \cup \left\{ \frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_i) : j \in \{0, 1, \ldots, \ell\} \right\} \cup \left\{ \frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma_{i+1}) : j \in \{1, 2, \ldots, \ell+1\} \right\}$$

modulo $\langle \overline{w} \rangle^\ell$. In going from i to $(i+1)$, the size of the generating set increases to $\tau_{i+1} \leqslant \tau_i + 2(\ell+1)$, which is at most $2(\ell+1)(i+1)$ by induction. Furthermore, each of the new elements of this generating set is again of the form $\frac{\partial P}{\partial z^j}(T, \overline{w}, \gamma)$ for some $j \leqslant (\ell+1)$ and $\gamma \in \mathbb{F}[T]$. □

An immediate consequence of Lemma 5.6.3 is the following corollary.

Corollary 5.6.6. *Let $k \in \mathbb{N}$, $P(T, \overline{w}, z) \in \mathbb{F}[T, \overline{w}, z]$ and $R(T, \overline{w}) \in \mathbb{F}[T, \overline{w}]$ be polynomials that satisfy the hypothesis of Lemma 5.6.3.*

Then, for every $\ell \in \mathbb{N}$, there is an algebraic circuit $C_\ell \in \mathbb{F}[T, \overline{w}]$ of depth at most $2\Delta + O(1)$ and size at most $\left(\mathrm{poly}(s, \deg(P)) \cdot (\log k)^{\mathrm{poly}(\ell)}\right)$ such that

$$C_\ell(T, \overline{w}) \equiv P\left(T, \overline{w}, R(T, \overline{w})\right) \text{ trunc } \langle \overline{w} \rangle^\ell .$$

5.6.2.1 *Why do we need the quadratic convergence version of Newton iteration?*

It would be instructive to step back and see why Newton iteration with quadratic convergence was required in the above argument. A slight modification of Lemma 5.4.7 will let us argue that low-degree components of approximate roots of a polynomial have small constant-depth circuits. However, there are subtle differences in the statement of Lemma 5.4.7 and the statement of Lemma 5.6.3. The key difference is that the approximate root ϕ_k is with respect to the variable T, whereas we are extracting homogeneous components of degree up to ℓ with respect to a *different set of variables*. Therefore, we may have contributions of $\overline{w}$-degree at most ℓ from terms with T-degree up to k. So, we cannot replace ϕ_k by a root of lower accuracy such as ϕ_ℓ.

In the above proof of Lemma 5.6.3, each level of the iteration adds to the number of "generators" used in the composition. Using the standard Newton iteration of k steps for φ_k would result in $O(k)$ generators and would not yield the required size bounds needed for our proof. The version of Newton iteration with quadratic convergence ensures that we obtain ϕ_k from $\log k$ iterations, and results in

the eventual number of "generators" as $O(\log k)$ instead (which was crucial for the final circuit size bound).

5.7 WARM-UP: PRESERVING TRUE ROOTS UNDER VARIABLE REDUCTION

In this section, we use the techniques of Section 5.6 to show a variable reduction map that will help us identify whether a given truncated power series root of a constant-depth circuit is a true root (and not just a sufficiently good truncation of a power series root). As described in the proof overview, the idea is to work with a PIT instance formed by plugging-in a candidate root into the input polynomial. In Lemma 5.7.1, we show that low-degree roots of such PIT instances have relatively small constant-depth circuits when the input polynomial has a small constant-depth circuit.

Lemma 5.7.1. *Let $k > 1$ be any natural number and let $P(T, \overline{w}, y, z) \in \mathbb{F}[T, \overline{w}, z]$ and $R(T, \overline{w}, y) \in \mathbb{F}[T, \overline{w}, y]$ be polynomials satisfying the following properties.*

- *$P(T, \overline{w}, y, z)$ is computable by a depth Δ circuit of size s.*
- *$P(T, \overline{w}, y, z)$ is T-regularized and monic in z.*
- *$R(T, \overline{w}, y)$ is a truncated, non-degenerate, approximate z-root of $P(T, \overline{w}, y, z)$ of order k with respect to T.*

Let $F(\overline{w}) \in \mathbb{F}[\overline{w}]$ be a polynomial of degree at most ℓ such that $(y - F(\overline{w}))$ divides $P(T, \overline{w}, y, R(T, \overline{w}, y))$. Then, F can be computed by an algebraic circuit of depth $(\Delta + O(1))$ and size at most

$$\mathrm{poly}(s, \deg(P)) \cdot (\log k)^{\mathrm{poly}(\ell)} .$$

Proof of Lemma 5.7.1. We start by setting ourselves up to invoke Lemma 5.4.8, a version of Lemma 5.4.7 from [CKS19]. To do this, we first need to ensure that the hypothesis of the lemma holds in our case.

SETTING UP TO INVOKE LEMMA 5.4.8: The first issue is that $F(\overline{w})$ could be a y-root of high multiplicity of $P(T, \overline{w}, y, R(T, \overline{w}, y))$. To work around this, we work with an appropriately high order derivative of P with respect to y. To this end, we start by viewing $P(T, \overline{w}, y, R(T, \overline{w}, y))$ as a univariate polynomial in y with coefficients from the ring $\mathbb{F}[T, \overline{w}]$. Thus, $P(T, \overline{w}, y, R(T, \overline{w}, y))$ can be decomposed as

$$P(T, \overline{w}, y, R(T, \overline{w}, y)) = \sum_{i=0}^{D} P_i(T, \overline{w}) y^i ,$$

where each P_i is a polynomial in only T and $\overline{w}$ variables. Let $(m+1)$ be the largest integer such that $(Y - F(\overline{w}))^{m+1}$ divides $P(T, \overline{w}, y, R(T, \overline{w}, y))$. So, to reduce the multiplicity of the y-root $F(\overline{w})$ to 1, we consider the polynomial $\hat{P}$ defined as

$$\hat{P}(T, \overline{w}, y) := \frac{\partial^m P(T, \overline{w}, y, R(T, \overline{w}, y))}{\partial y^m}.$$

Thus, we have that $(y - F(\overline{w}))$ divides $\hat{P}$ and does not divide $\frac{\partial \hat{P}}{\partial y}$. As a consequence, we get that $\frac{\partial \hat{P}}{\partial y}(T, \overline{w}, F(\overline{w}))$ is a non-zero polynomial in $\mathbb{F}[T, \overline{w}]$. By shifting the $\overline{w}$ variables if needed to $\overline{a} + \overline{w}$ for some $\overline{a} \in \mathbb{F}^{|\overline{w}|}$, we get that

$$\frac{\partial \hat{P}}{\partial y}(T, \overline{0}, F(\overline{0})) \not\equiv 0.$$

We are now in a position to invoke Lemma 5.4.8, which tells us that there exists a $\kappa \in \mathbb{F}$ and a $(\ell+1)$-variate polynomial $Q(\overline{u}) \in \mathbb{F}[\overline{u}]$ of degree at most ℓ such that

$$F(\overline{w}) \equiv Q(h_0(\kappa, \overline{w}), h_1(\kappa, \overline{w}), \ldots, h_\ell(\kappa, \overline{w})) \mod \langle \overline{w} \rangle^{\ell+1}, \tag{5.7.2}$$

where for every $j \in \{0, 1, \ldots, \ell\}$, $h_i(T, \overline{w})$ is defined as

$$h_j(T, \overline{w}) := \left(\frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{w}, F(\overline{0})) - \frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{0}, F(\overline{0})) \right) \text{ trunc } \langle \overline{w} \rangle^{\ell+1}.$$

Given the bounds on the number of variables and $\overline{u}$-degree of $Q(\overline{u})$, we get that $Q(\overline{u})$ is computable by an algebraic circuit of depth-2 and size at most $\exp(O(\ell))$. At this point, if we can somehow replace each $h_i(T, \overline{w})$ by a small constant-depth circuit, we can obtain a small constant-depth circuit for $F(\overline{w})$ by combining these with the depth-2 circuit for Q, and then extracting certain homogeneous components of interest.

CONSTANT-DEPTH CIRCUITS FOR $h_i(T, \overline{w})$: Recall that $\hat{P}(T, \overline{w}, y)$ is a derivative of $P(T, \overline{w}, y, R(T, \overline{w}, y))$ with respect to y^m. Thus, from Corollary 5.3.2, we get that $\hat{P}$ can be written as an $\mathbb{F}[y]$-weighted linear combination of the polynomials in the set

$$\{P(T, \overline{w}, \beta_i, R(T, \overline{w}, \beta_i)) : i \in \{0, 1, \ldots, \deg(P)\}\}$$

where β_i's are distinct field constants and every $\mathbb{F}[y]$-weight $\Lambda_{j,i}$ in the linear combination for $\frac{\partial}{\partial y^{j+m}} P(T, \overline{w}, y, R(T, \overline{w}, y))$ has y-degree at most $\deg(P)$.

$$\begin{aligned}
h_j(T, \overline{w}) &= \left(\frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{w}, F(\overline{0})) - \frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{0}, F(\overline{0})) \right) \text{ trunc } \langle \overline{w} \rangle^{\ell+1} \\
&= \left(\left(\frac{\partial}{\partial y^{j+m}} P(T, \overline{w}, y, R(T, \overline{w}, y)) \right) \Bigg|_{y=F(\overline{0})} - \frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{0}, F(\overline{0})) \right) \text{ trunc } \langle \overline{w} \rangle^{\ell+1} \\
&= \left(\left(\sum_{i=0}^{D} \Lambda_{j,i}(F(\overline{0})) P(T, \overline{w}, \beta_i, R(T, \overline{w}, \beta_i)) \right) - \frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{0}, F(\overline{0})) \right) \text{ trunc } \langle \overline{w} \rangle^{\ell+1} \\
&= \left(\sum_{i=0}^{D} \Lambda_{j,i}(F(\overline{0})) P(T, \overline{w}, \beta_i, R(T, \overline{w}, \beta_i)) \text{ trunc } \langle \overline{w} \rangle^{\ell+1} \right) - \frac{\partial^j \hat{P}}{\partial y^j}(T, \overline{0}, F(\overline{0}))
\end{aligned}$$

Thus, to show that $h_i(T, \overline{w})$ have small constant-depth circuits, it suffices (up to multiplicative factors of $\text{poly}(\deg(P))$) to show that the polynomials $(P(T, \overline{w}, \beta_i, R(T, \overline{w}, \beta_i)) \text{ trunc } \langle \overline{w} \rangle^{\ell+1})$ have small constant-depth circuits. But, this is exactly the content of Corollary 5.6.6, which[6] shows that $(P(T, \overline{w}, \beta_i, R(T, \overline{w}, \beta_i)) \text{ trunc } \langle \overline{w} \rangle^{\ell+1})$ has a circuit of depth $\Delta + O(1)$ and size

$$\left(\text{poly}(s, \deg(P)) \cdot (\log k)^{\text{poly}(\ell)} \right).$$

This gives a circuit $C_i(T, \overline{w})$ of roughly the same size with a constant additive increase in depth for each $h_i(T, \overline{w})$, and thus a circuit $\tilde{C}_i(\overline{w}) := C_i(\kappa, \overline{w})$ for each $h_i(\kappa, \overline{w})$, where κ is the field constant in Eq. (5.7.2).

PUTTING THINGS TOGETHER: Plugging in these circuits as inputs to the depth-2 circuit for $Q(\overline{u})$, we get that there is a circuit $C(\overline{w}) = Q(\tilde{C}_0(\overline{w}), \ldots, \tilde{C}_\ell(\overline{w}))$ of size $\left(\text{poly}(s, \deg(P)) \cdot (\log k)^{\text{poly}(\ell)} \right)$ and depth $(\Delta + O(1))$ such that

$$F(\overline{w}) \equiv C(\overline{w}) \bmod \langle \overline{w} \rangle^{\ell+1}.$$

The $\overline{w}$-degree of $C(\overline{w})$ is at most $\ell \cdot \deg(Q)$, which is at most ℓ^2. By using Corollary 5.3.2 to compute the truncation $C(\overline{w}) \text{ trunc } \langle \overline{w} \rangle^{\ell+1}$, we get a depth-$(\Delta + O(1))$ circuit for $F(\overline{w})$, of size at most

$$\left(\text{poly}(s, \deg(P)) \cdot (\log k)^{\text{poly}(\ell)} \right).$$

□

Now, we will use Lemma 5.7.1 to prove the main theorem of this section: the Kabanets-Impagliazzo generator, instantiated with an appropriate design and a low-degree polynomial, preserves the roots

6 There is a slight subtlety here: $P(T, \overline{w}, y, R(T, \overline{w}, y))$ must still satisfy the hypothesis of Lemma 5.6.3/Corollary 5.6.6 after replacing each y by a field constant β_i. But this is true because of Lemma 5.6.1.

of constant-depth circuits, while ensuring that no new roots are created. This is a special case of Theorem 5.5.1.

Theorem 5.7.3 (Checking validity of a root)**.** *Let* $k > 1$ *be any natural number and let* $A(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ *and* $\Phi(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ *be polynomials satisfying the following properties.*

- $A(T, \overline{x}, z)$ *is computable by a depth* Δ *circuit of size* s*.*
- $A(T, \overline{x}, z)$ *is* T*-regularized and monic in* z*.*
- $\Phi(T, \overline{x})$ *is a truncated, non-degenerate, approximate* z*-root of* $A(T, \overline{x}, z)$ *of order* k
- $A(T, \overline{x}, \Phi(T, \overline{x}))$ *is not* identically zero *in* $\mathbb{F}[T, \overline{x}]$*.*

Let $\mathcal{S}$ *be a* (n, σ, μ, ρ)*-design and let* g *be a* σ*-variate polynomial of degree* d*. For a new tuple* $\overline{w}$ *of* μ *variables distinct from* $T, \overline{x}, z$*, let us define the polynomial map* $\mathbf{KI}_{g,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ *using Definition 5.3.9. Then, the following statement is true.*

If $A(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \Phi(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w})))$ *is identically zero, then* g *can be computed by an algebraic circuit of depth* $\Delta' = (\Delta + O(1))$ *and size* s' *which is at most*

$$\mathrm{poly}(s, \deg(A), k) \cdot (\rho \log k)^{\mathrm{poly}(d)} .$$

In particular, if g *cannot be computed by a size* s' *depth* Δ' *circuit, then* $A(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \Phi(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w})))$ *is identically zero if and only if* $A(T, \overline{x}, \Phi(T, \overline{x}))$ *is identically zero.*

The proof of this theorem is based on some of the same principles as that of the proof of Theorem 5.5.1. However, the underlying technical details are considerably shorter and cleaner. We now discuss the proof.

Proof of Theorem 5.7.3. Let $\hat{B}_j(T, \overline{x}, \overline{w})$ be the polynomial obtained by replacing the variables $x_1, \ldots, x_{j-1}$ in the polynomial $A(T, \overline{x}, \Phi(T, \overline{x}))$ by the polynomials $g_1, \ldots, g_{j-1} \in \mathbb{F}[\overline{w}]$ respectively. From the hypothesis of the theorem, we know that $\hat{B}_0$ is not identically zero and $\hat{B}_n$ is identically zero. Thus, there must be a j such that $\hat{B}_j$ is not identically zero but $\hat{B}_{j+1}$ is identically zero. We focus on this j for the rest of the proof.

Note that $\hat{B}_j$ depends on the variables $T, \overline{w}$ and $x_j, x_{j+1}, \ldots, x_n$ but does not depend on the variables $x_1, x_2, \ldots, x_{j-1}$. Moreover, $\hat{B}_{j+1}$ is obtained from $\hat{B}_j$ by substituting x_j by g_j. For ease of notation, we refer to the variable x_j as y for the rest of this argument. Let B_j be the polynomial obtained from $\hat{B}_j$ by setting the variables $x_{j+1}, \ldots, x_n$ and $\{w_i : w_i \notin S_j\}$ to constants from $\mathbb{F}$ such that B_j remains non-zero. Since $\hat{B}_j$ is a non-zero polynomial, a random substitution of these variables from $\mathbb{F}$ (assuming it is large enough) has this property. Thus, B_j only depends on the variables $T, y, \{w_i : w_i \in S_j\}$. For ease of notation, we continue to refer to the tuple $\overline{w}_{S_j}$ as $\overline{w}$.

We know that when we substitute y by g_j in B_j, we end up with the identically zero polynomial. In other words, $(y - g_j)$ divides B_j. At this point, we would like to invoke Lemma 5.7.1 to conclude the proof. In order to do that, we need to make sure that the hypothesis of the lemma holds, which we do now.

From its definition, we know that

$$\hat{B}_j(T, \overline{x}, \overline{w}) := A(T, g_1(\overline{w}), \ldots, g_{j-1}(\overline{w}), x_j, x_{j+1}, \ldots, x_n) .$$

Now, to obtain B_j from $\hat{B_j}$, we set the variables $x_{j+1}, \ldots, x_n$ and the $\overline{w}$ variables outside the set S_j to field constants, and rename x_j as y. For every $i < j$, the above setting of $\overline{w}$ variables outside the set S_j to field constants reduces each g_i to a polynomial $\hat{g}_i$ that has degree at most d and only depends on at most ρ $\overline{w}$ variables in the set $|S_i \cap S_j|$. Thus, $\hat{g}_i$ can be written as a sum of at most $\binom{\rho+d}{d} \leqslant \rho^d$ many monomials, and hence is a depth 2 algebraic circuit of size at most $\rho^{O(d)}$.

For $i \in [j+1, n]$, let the variable x_i be set to the field element α_i, and let $\hat{A}(T, \overline{w}, y)$ and $\hat{\Phi}(A, \overline{w}, y)$ be the polynomials defined as

$$\hat{A}(T, \overline{w}, y, z) := A\left(T, \hat{g}_1(\overline{w}), \ldots, \hat{g}_{j-1}(\overline{w}), y, \alpha_{j+1}, \ldots, \alpha_n, z\right) ,$$

and

$$\hat{\Phi}(T, \overline{w}, y) := \Phi\left(T, \hat{g}_1(\overline{w}), \ldots, \hat{g}_{j-1}(\overline{w}), y, \alpha_{j+1}, \ldots, \alpha_n\right) .$$

Thus, we also get that $B_j(T, \overline{w}, y) = \hat{A}(T, \overline{w}, y, \hat{\Phi})$. A direct application of Lemma 5.6.1 tells us that $\hat{A}(T, \overline{w}, y, z)$ is T-regularized and monic in z, and $\hat{\Phi}$ is a truncated, non-degenerate, approximate z-root of $\hat{A}(T, \overline{w}, y, z)$ of order k. Also, since the degree of each $\hat{g}_i$ is at most d, we have that the total degree of $\hat{A}$ is at most $d \cdot \deg(A)$. Since A has a circuit of depth Δ and size s, from the discussion above, we can plug in a small depth-2 circuit of size at most $\rho^{O(d)}$ for each $\hat{g}_i$, and get that $\hat{A}$ has a circuit of depth $\Delta + 2$ and size $\left(s\rho^{O(d)}\right)$.

So, the hypothesis of Lemma 5.7.1 continue to hold if we set the polynomials P and R in Lemma 5.7.1 to $\hat{A}$ and $\hat{\Phi}$ respectively, with size and depth bound on the circuit complexity of $\hat{A}$ being $\left(s\rho^{O(d)}\right)$ and $\Delta + 2$ respectively.

Finally, since $(y - g_j(\overline{w}))$ divides $B_j(T, \overline{w}, y)$, which equals $(\hat{A}(T, \overline{w}, y, \hat{\Phi}(T, \overline{w}, y)))$, we get from an application of Lemma 5.7.1 that $g_j(\overline{w})$ has a depth $\Delta' = (\Delta + O(1))$ circuit of size s' which is at most

$$\left(\mathrm{poly}(s\rho^d, \deg(\hat{A}), k) \cdot (\log k)^{\mathrm{poly}(d)}\right) \leqslant \left(\mathrm{poly}(s, \deg(A), k) \cdot (\rho \log k)^{\mathrm{poly}(d)}\right) .$$

The contrapositive tells us that if every depth-Δ' circuit for g requires size greater than s' obtained as an upper bound above, then $A(T, \mathbf{KI}_{g,S}(\overline{w}), \Phi(T, \mathbf{KI}_{g,S}(\overline{w})))$ is identically zero if and only if

$A(T, \overline{x}, \Phi(T, \overline{x}))$ is identically zero. This completes the proof of the theorem. □

5.8 PRESERVING IRREDUCIBILITY UNDER VARIABLE REDUCTION

We prove our main technical result (Theorem 5.5.1) in this section. Together with the ideas discussed in Section 5.9, this is sufficient to complete the proof of correctness of our algorithms. To this end, we first start by showing that the irreducibility of factors of multivariate polynomials can be characterized by certain divisibility tests involving approximate power series roots of the polynomial. We then show that these divisibility tests can be reduced to polynomial identity tests for certain circuits, where the instances are built using constant-depth circuits and their approximate power series roots. A reduction from divisibility testing to PIT was shown by Forbes [For15] and was later used in factorisation algorithms in [KRS23; Kum+24]. However, our proof for this reduction here is different from that in [For15] and goes via a recent result of Andrews & Wigderson [AW24] that shows that the transformation between elementary symmetric polynomials and power symmetric polynomials can be done efficiently using constant-depth circuits. This alternative proof offers a clearer insight into the structure of the PIT instances and this structure is helpful in completing our proof. Finally, we show that these PIT instances can be solved deterministically in subexponential time using the Kabanets-Impagliazzo generator invoked with the explicit low degree hard polynomials in [LST25]. This analysis is the core technical part of the proof and is perhaps a little surprising since we are unable to show that the PIT instances we work with are themselves constant-depth circuits. Nevertheless, we manage to show that the generator can still be analyzed and shown to work for such circuits.

5.8.1 From irreducibility testing to divisibility tests

Throughout this section, we assume that $P(T, \overline{x}, z)$ is a nonzero squarefree polynomial that is monic in z, T-regularised with respect to z, and it satisfies that $P(0, \overline{x}, z) = P(0, \overline{0}, z)$ is squarefree. Let $D = \deg_z P$ and let $P(T, \overline{x}, z) = \prod_{i \in [D]} (z - \phi_i(T, \overline{x}))$ be the factorisation[7] of $P(T, \overline{x}, z)$ where $\phi_i(T, \overline{x})$ are from the ring of power series $\overline{\mathbb{F}}[\overline{x}][\![T]\!]$.

7 Such a factorisation exists because each root of the univariate $P(0, \overline{x}, z)$ can be extended to a power series root via Newton iteration; see [DSS22, Theorem 4].

Lemma 5.8.1. *Let* $P(T, \overline{x}, z)$ *and* $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ *be defined as above. For any* $G(T, \overline{x}, y) \in \overline{\mathbb{F}}[T, \overline{x}, z]$ *that divides* $P(T, \overline{x}, z)$*, there is some subset* $U \subseteq [D]$ *such that*

$$G(T, \overline{x}, y) = \prod_{i \in U} (z - \phi_i(T, \overline{x})).$$

Proof sketch. Since $G(T, \overline{x}, z)$ is a factor of $P(T, \overline{x}, z)$, it must also be monic in z, squarefree, and have the property that $G(0, \overline{0}, z)$ is squarefree. Thus, Newton iteration, which gives us the factorisation $P(T, \overline{x}, z) = \prod_{i \in [D]} (z - \phi_i(T, \overline{x}))$ (see Footnote 7), also shows that $G(T, \overline{x}, z)$ must be $\prod_{i \in U} (z - \phi_i(T, \overline{x}))$ for some $U \subseteq [D]$. □

It will be convenient to have the following definition for the set of *candidate factors* of $P(T, \overline{x}, y)$.

Definition 5.8.2 (Candidate factors)**.** *Let* $P(T, \overline{x}, z)$ *and* $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ *be defined as above. We define the set*

$$\left\{ \prod_{i \in U} (z - \phi_i(T, \overline{x})) \text{ trunc } T^k : U \subseteq [D] \right\} \subseteq \overline{\mathbb{F}}[T, \overline{x}, z]$$

as the set of candidate factors *of* P. ◊

For our algorithm, we would be interested in divisibility over $\mathbb{F}$ and not over $\overline{\mathbb{F}}$. The following lemma gives some sufficient conditions for when a candidate factor $\prod_{i \in U} (z - \phi_i(T, \overline{x}))$ trunc $T^k \in \overline{\mathbb{F}}[T, \overline{x}, z]$ has coefficients in $\mathbb{F}$.

Lemma 5.8.3. *Let* $P(T, \overline{x}, z)$ *be a polynomial and* $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ *be power series as defined above. For each* $i \in [D]$*, let* $\alpha_i = \phi_i(0, \overline{0})$*, and let* $\mathcal{Z} = \{\alpha_1, \ldots, \alpha_D\}$ *(which are the roots of* $P(0, \overline{x}, z) = P(0, \overline{0}, z)$ *in the algebraic closure* $\overline{\mathbb{F}}$*). Fix some* $k \geqslant 0$*. For a subset* $S \subset [D]$*, let*

$$Q_S(T, \overline{x}, z) := \prod_{i \in S} (z - \phi_i(T, \overline{x})) \text{ trunc } T^k.$$

If the polynomial $g_S(z) = \prod_{i \in S} (z - \alpha_i) = Q_S(0, \overline{0}, z) = Q_S(0, \overline{x}, z)$ *has all its coefficients in the base field* $\mathbb{F}$*, then so does* $Q_S(T, \overline{x}, z)$*.*

Proof. Define the following rational functions $\Gamma^{(i)}(T, \overline{x}, \zeta)$ for $i \geqslant 0$ as follows:

$$\Gamma^{(0)}(\zeta) := \zeta$$
$$\Gamma^{(i+1)}(\zeta) := \Gamma^{(i)}(\zeta) - \frac{P(T, \overline{x}, \Gamma^{(i)}(\zeta))}{\partial_z P(0, \overline{0}, \zeta)} \quad \text{for } i > 0.$$

Note that $\Gamma^{(i)}(\alpha)$ is in fact a polynomial in $T, \overline{x}$ for every α such that $\partial_z P(0, \overline{0}, \alpha) \neq 0$. By Lemma 5.4.3, for any $\alpha_i \in \mathcal{Z}$, we have that $\Gamma^{(k)}(\alpha_i) = \phi_i \bmod T^k$ and therefore

$$Q_S(T, \overline{x}, z) = \prod_{i \in S} (z - \Gamma^{(k)}(\alpha_i)) \text{ trunc } T^k.$$

Therefore, every coefficient of $Q_S(T, \overline{x}, z)$ is a symmetric function of the set $\{\alpha_i : i \in S\}$. Since $g(z) = \prod_{i \in S}(z - \alpha_i)$ is assumed to have coefficients in the base field $\mathbb{F}$, this implies that every elementary symmetric function of the set $\{\alpha_i : i \in S\}$ is in $\mathbb{F}$. By the fundamental theorem of symmetric polynomials, this implies that every coefficient of $Q_S(T, \overline{x}, z)$ is also in $\mathbb{F}$. □

With the above two lemmas, we can now characterise the factors (over $\mathbb{F}$) of $P(T, \overline{x}, z)$ using the polynomials $Q_S(T, \overline{x}, z)$ defined above.

Lemma 5.8.4. *Let $P(T, \overline{x}, z)$ and $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ be defined as above, and let $k > \deg_T P$. Let $\alpha_i = \phi_i(0, \overline{0})$ for each $i \in [D]$. For a subset $S \subseteq [D]$, define the polynomial*

$$Q_S(T, \overline{x}, z) = \prod_{i \in S} (z - \phi_i(T, \overline{x})) \text{ trunc } T^k.$$

Then, the set of factors of P over the base field $\mathbb{F}$ is the same as

$$\left\{ Q_U(T, \overline{x}, z) \in \overline{\mathbb{F}}[T, \overline{x}, z] \ : \ \begin{array}{c} U \subseteq [D] \text{ with } Q_U(T, \overline{x}, z) \text{ dividing } P \\ \text{and } Q_U(0, \overline{x}, z) = Q_U(0, \overline{0}, z) \in \mathbb{F}[z] \end{array} \right\}$$

Proof. Any $G(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ that is a factor of $P(T, \overline{x}, z)$, by Lemma 5.8.1, is equal to $Q_U(T, \overline{x}, z) = \prod_{i \in U}(z - \phi_i(T, \overline{x}))$ for some $U \subseteq [D]$. Additionally, if $G(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$, then $G(0, \overline{x}, z) = Q_U(0, \overline{x}, z) \in \mathbb{F}[z]$.

For the other direction, consider any $Q_U(T, \overline{x}, z) \in \overline{\mathbb{F}}[T, \overline{x}, z]$ that divides $P(T, \overline{x}, z)$ and also satisfies that $Q_U(0, \overline{0}, z) = \prod_{i \in U}(z - \alpha_i)$ is a polynomial with coefficients in $\mathbb{F}$. By Lemma 5.8.3, we have that $Q_U(T, \overline{x}, z)$ must also be a polynomial in $\mathbb{F}[T, \overline{x}, z]$ and hence is a factor of $P(T, \overline{x}, z)$ over $\mathbb{F}$. □

This immediately implies the following *certificate* for irreducibility.

Corollary 5.8.5. *Let $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ and $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ be defined as above, and let $k > \deg_T(P)$. Then $P(T, \overline{x}, z)$ is irreducible over $\mathbb{F}$ if and only if for every $\emptyset \neq S \subsetneq [D]$ such that $\prod_{i \in S}(z - \alpha_i)$ has all coefficients in $\mathbb{F}$, we have that $Q_S(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ does not divide $P(T, \overline{x}, z)$ where*

$$Q_S(T, \overline{x}, z) = \prod_{i \in S} (z - \phi_i(T, \overline{x})) \text{ trunc } T^k.$$

Essentially the same proof as Lemma 5.8.4 also characterizes the set of factors of $P(T, \overline{x}, z)$ over the algebraic closure $\overline{\mathbb{F}}$, and thus gives a

certificate for *absolute irreducibility* using divisibility tests over $\overline{\mathbb{F}}$, as stated in the following corollary.

Corollary 5.8.6. *Let* $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ *and* $\phi_1(T, \overline{x}), \ldots, \phi_D(T, \overline{x})$ *be defined as above, and let* $k > \deg_T(P)$. *Then* $P(T, \overline{x}, z)$ *is absolutely irreducible if and only if for every* $\emptyset \neq S \subsetneq [D]$, *we have that* $Q_S(T, \overline{x}, z) \in \overline{\mathbb{F}}[T, \overline{x}, z]$ *does not divide* $P(T, \overline{x}, z)$ *where*

$$Q_S(T, \overline{x}, z) = \prod_{i \in S} (z - \phi_i(T, \overline{x})) \text{ trunc } T^k.$$

Corollary 5.8.5 together with the reduction from divisibility testing to PIT in the next section gives us a reduction from irreducibility testing to PIT. However, we note that Corollary 5.8.5 gives exponentially many (in degree D) divisibility testing instances, and thus it isn't immediately clear if this reduction can be useful for obtaining subexponential time irreducible testing algorithms. It is for this reason that our algorithm for factorisation does not proceed directly using this reduction and only uses the corollary (and the previous lemmas) in its analysis.

5.8.2 From divisibility testing to polynomial identity tests

Let us recall the standard definitions of the power sum symmetric polynomials and the elementary symmetric polynomials, particularly in the context of roots of a univariate polynomial.

Definition 5.8.7. *For any monic univariate polynomial* $P = \prod_{j \in [d]} (z - \alpha_j) \in \mathbb{F}[z]$ *and natural number* i, $\mathbf{Psym}_i(P)$ *and* $\mathbf{Esym}_i(P)$ *are the power sum symmetric polynomial* $\sum_j \alpha_j^i$ *and the elementary symmetric polynomial* $\sum_{S \subseteq [d] : |S| = i} \prod_{j \in S} \alpha_j$ *of degree* i *respectively, in the multiset of roots of* P. ◊

The following important lemma from a recent beautiful work of Andrews and Wigderson [AW24] (but also present in earlier works such as [SW01]) shows that there is a constant-depth circuit that computes the power sum symmetric polynomials of a multiset from the elementary symmetric polynomials of the multiset; similarly, there is a constant-depth circuit that computes the elementary symmetric polynomials of a multiset from the power sum symmetric polynomials of the multiset.

Lemma 5.8.8 ([AW24, Lemma 3.4, Lemma 3.6])**.** *Let* $\mathbb{F}$ *be any field of characteristic zero and* $n \in \mathbb{N}$ *be any natural number. Then, there is a constant-depth circuit of size and degree* $\mathrm{poly}(n)$ *that takes the* n*-variate polynomials* $\{\mathbf{Esym}_i(\overline{x}) : i \in [n]\}$ *as inputs and outputs the polynomials* $\{\mathbf{Psym}_i(\overline{x}) : i \in [n]\}$.

Similarly, there is a constant-depth circuit of size and degree $\mathrm{poly}(n)$ *that takes the* n*-variate polynomials* $\{\mathbf{Psym}_i(\overline{x}) : i \in [n]\}$ *as inputs and outputs the polynomials* $\{\mathbf{Esym}_i(\overline{x}) : i \in [n]\}$.

In [For15, Section 7.2], Forbes showed that the question of deterministic divisibility testing for polynomials from a certain complexity class can be reduced to deterministic PIT for polynomials from a related complexity class. This reduction was extensively used in [KRS23] and [Kum+24]. The following version offers a different approach for reducing divisibility testing to PIT, using Lemma 5.8.8.

Lemma 5.8.9. *Let $D \geqslant t \geqslant 0$ be integer parameters. Let $\mathbb{F}$ be any field of characteristic zero. Then, there is a constant-depth $\mathrm{poly}(D,t)$-sized circuit $\mathrm{DivTest}_{D,t}$ on $D+t+1$ variables, that takes $(D+t)$ inputs labelled $f_0,\ldots,f_{D-1} \in \mathbb{F}$ and $g_0,\ldots,g_{t-1} \in \mathbb{F}$ respectively, such that*

$$\mathrm{DivTest}_{D,t}(z, f_0,\ldots,f_{D-1}, g_0,\ldots,g_{t-1}) = 0$$

if and only if the polynomial $f(z) = f_0 + f_1 z + \cdots + f_{D-1}z^{D-1} + z^D$ divides the polynomial $g(z) = g_0 + g_1 z + \cdots + g_{t-1}z^{t-1} + z^t$.

Proof. By Lemma 5.8.8, we can construct multi-output circuits $\mathrm{PSymToESym}_n$ and $\mathrm{ESymToPSym}_n$ of size $\mathrm{poly}(n)$ and depth $O(1)$ such that for every choice of $\alpha_1,\ldots,\alpha_n$ we have

$$\begin{aligned}
\mathrm{ESymToPSym}_n(E_1,\ldots,E_n) &= (P_1,\ldots,P_n)\\
\mathrm{PSymToESym}_n(P_1,\ldots,P_n) &= (E_1,\ldots,E_n)\\
\text{where } E_i &= \mathbf{Esym}_i(\alpha_1,\ldots,\alpha_n)\\
\text{and } P_i &= \mathbf{Psym}_i(\alpha_1,\ldots,\alpha_n).
\end{aligned}$$

Let $f(z) = \prod_{i=1}^{D}(z-\alpha_i)$ where $S_f = \{\alpha_1,\ldots,\alpha_D\}$ is the multiset of roots of f (in the algebraic closure of $\mathbb{F}$), and let S_g be the multiset of roots of $g(z)$.

Note that $e_i^{(f)} := \mathbf{Esym}_i(S_f) = (-1)^i f_{D-i}$ for $i = 1,\ldots,D$ and $e_i^{(g)} = \mathbf{Esym}_i(S_g) = (-1)^i g_{t-i}$ for $i = 1\ldots,t$. Define the *pseudo-quotient* of f and g, referred to by $\tilde{h}$ as follows:

$$\begin{aligned}
&\tilde{h}(z) = z^{D-t} - z^{D-t-1}e_1^{(h)} + \cdots + (-1)^{D-t}e_{D-t}^{(h)}\\
&\text{where } e_i^{(h)} = \mathrm{PSymToESym}_{D-t}(r_1,\ldots,r_{D-t})\\
&\quad\text{where } r_j = \left(\mathrm{ESymToPSym}_D\left(e_1^{(f)},\ldots,e_D^{(f)}\right) - \mathrm{ESymToPSym}_t\left(e_1^{(g)},\ldots,e_t^{(g)}\right)\right)_j
\end{aligned}$$

If $g \mid f$, then the multiset S_f is the union of the multiset S_g and the multiset S_h of roots of $h = f/g$. In that case, for each $i = 1,\ldots,D-t$, we have

$$\begin{aligned}
&\mathbf{Psym}_i(S_f) = \mathbf{Psym}_i(S_g) + \mathbf{Psym}_i(S_h)\\
\implies &\mathbf{Psym}_i(S_h) = \mathbf{Psym}_i(S_f) - \mathbf{Psym}_i(S_g).
\end{aligned}$$

Therefore, $\tilde{h}(z) = h(z)$ if $g \mid f$.

The circuit $\mathrm{DivTest}_{D,t}(f_0, \ldots, f_{D-1}, g_0, \ldots, g_{t-1})$ outputs the polynomial $f(z) - \tilde{h}(z) \cdot g(z)$ where $\tilde{h}(z)$ is computed as stated above. By construction, this is a circuit of size $\mathrm{poly}(D, t)$ and depth $O(1)$. Furthermore, as argued above, if $g \mid h$ then $\tilde{h}(z) = h(z) = f(z)/g(z)$ and hence the above computes the zero polynomial. If $g(z) \nmid f(z)$, then $f(z) - \tilde{h}(z)g(z)$ is nonzero for every choice of $\tilde{h}(z)$ and hence the above is nonzero polynomial. □

The following lemma uses Lemma 5.8.9 to show that we can reduce the question of checking whether a candidate factor (constructed using various approximate roots from Newton iteration) is an actual factor, to a PIT instance.

Lemma 5.8.10. *Let $\mathbb{F}$ be a field of characteristic zero. Let $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ be a polynomial that is monic in z. Let $R_1(T, \overline{x}), R_2(T, \overline{x}), \ldots, R_\ell(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ be polynomials of T-degree at most $(k-1)$.*

Then, the constant-depth circuit $\mathrm{DivTest}_{\deg_z(P),\ell}$ (from Lemma 5.8.9) on at most $(\deg_z(P) + \ell + 1)$-variables and size and degree at most $\mathrm{poly}(\deg(P), \ell, k)$, has the property that

$$Q(T, \overline{x}, z) := \left(\prod_{i \in [\ell]} (z - R_i(T, \overline{x})) \right) \text{ trunc } T^k$$

divides $P(T, \overline{x}, z)$ if and only if

$$\mathrm{DivTest}_{\deg_z(P),\ell}\Big(z, P_0(T, \overline{x}), \ldots, P_{\deg_z(P)}(T, \overline{x}), (\mathbf{Esym}_1(\mathcal{R}) \text{ trunc } T^k), \\ \ldots, (\mathbf{Esym}_\ell(\mathcal{R}) \text{ trunc } T^k)\Big) \equiv 0,$$

where, for every i, $P_i(T, \overline{x})$ is the coefficient of z^i in P, when viewing it as a univariate in z with coefficients in the ring $\mathbb{F}[T, \overline{x}]$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_\ell\}$.

Proof. This almost immediately follows from Lemma 5.8.9 since each of the polynomials
$\mathbf{Esym}_1(\mathcal{R})$ trunc $T^k, \ldots, \mathbf{Esym}_\ell(\mathcal{R})$ trunc T^k are the coefficients of $\left(\prod_{i \in [\ell]}(z - R_i(T, \overline{x}))\right)$ trunc T^k. As it is, the DivTest circuit would capture divisibility when the coefficients of the input (univariate) polynomials (in the variable z) are from a field, but since we are dealing with monic polynomials, Gauss' Lemma (Lemma 3.3.4) ensures that $Q(T, \overline{x}, z)$ divides $P(T, \overline{x}, z)$ in $\mathbb{F}(T, \overline{x})[z]$ if and only if $Q(T, \overline{x}, z)$ divides $P(T, \overline{x}, z)$ in $\mathbb{F}(T, \overline{x})[z]$. The theorem now follows. □

Lemma 5.8.11. *Let $\mathbb{F}$ be a field of characteristic zero. Let $R_1(T, \overline{x}), R_2(T, \overline{x}), \ldots, R_\ell(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ be polynomials of T-degree at most $(k-1)$. Let $\{\alpha_j : j \in [\mathrm{poly}(\ell, k)]\}$ be a sufficiently large set of arbitrary distinct elements of $\mathbb{F}$.*

Then, there is a multi-output constant-depth algebraic circuit $\hat{C}$ of size $\mathrm{poly}(\ell, k)$ that takes as input the polynomials of the form $\{R_i(\alpha_j T, \overline{x}) :$

$i \in [\ell], j \in [\mathrm{poly}(\ell, k)]\}$ *and outputs* $\left(\mathbf{Esym}_i(R_1, R_2, \ldots, R_\ell) \text{ trunc } T^k\right)$ *for every* $i \in [\ell]$.

Proof. From Lemma 5.8.8, we get that there is a multi-output constant-depth circuit C on ℓ variables that can be computed by a constant-depth circuit of size and degree $\mathrm{poly}(\ell)$ such that

$$C(\mathbf{Psym}_1(\mathcal{R}), \ldots, \mathbf{Psym}_n(\mathcal{R})) = (\mathbf{Esym}_1(\mathcal{R}), \ldots, \mathbf{Esym}_n(\mathcal{R})),$$

where, $\mathcal{R} = (R_1, R_2, \ldots, R_\ell)$. For ease of notation, we just focus on the computation of one of the $\mathbf{Esym}_j(\mathcal{R})$ and consider the equality

$$\mathbf{Esym}_j(\mathcal{R}) = C(\mathbf{Psym}_1(\mathcal{R}), \ldots, \mathbf{Psym}_n(\mathcal{R})).$$

The above equality immediately gives us a constant-depth circuit that takes $\mathcal{R}$ as input and outputs their elementary symmetric polynomials. However, the goal is to compute these things modulo T^k.

To recover $\mathbf{Esym}_j(\mathcal{R}) \text{ trunc } T^k$ from the above equality, we think of each $R_i(T, \overline{x})$ as a polynomial in $\mathbb{F}[\overline{x}][T]$ and apply Corollary 5.3.2 with respect to the variable T since we want to extract all the monomials that have T-degree at most k. This only incurs a polynomial blow up in size and an additive constant increase in depth. Moreover, the new circuit can be seen to be taking as inputs polynomials of the form $\{R_i(\alpha_j T, \overline{x}) : i \in [\ell], j \in [\mathrm{poly}(\ell, k)]\}$. □

The following is an immediate consequence of Lemma 5.8.11 and Lemma 5.8.10.

Corollary 5.8.12. *Let* $\mathbb{F}$ *be a field of characteristic zero. Let* $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ *be a polynomial that is monic in* z*. Let* $R_1(T, \overline{x}), R_2(T, \overline{x}), \ldots, R_\ell(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ *be polynomials of* T*-degree at most* $(k-1)$*. Let* $\{\alpha_j : j \in [\mathrm{poly}(\ell, k, \deg(P))]\}$ *be a sufficiently large set of arbitrary distinct elements of* $\mathbb{F}$*.*

Then, there exists a constant-depth circuit C *on at most* $(\mathrm{poly}(\ell, k, \deg(P)))$*-variables and size and degree at most* $\mathrm{poly}(\deg(P), \ell, k)$ *such that*

$$\left(\prod_{i \in [\ell]} (z - R_i(T, \overline{x}))\right) \text{ trunc } T^k$$

divides $P(T, \overline{x}, z)$ *if and only if*

$$C\left(z, P_0(T, \overline{x}), \ldots, P_{\deg_z(P)}(T, \overline{x}), \mathcal{R}\right) \equiv 0,$$

where, for every i, $P_i(T, \overline{x})$ *is the coefficient of* z^i *in* P*, when viewing it as a univariate in* z *with coefficients in the ring* $\mathbb{F}[T, \overline{x}]$ *and* $\mathcal{R} = \left(R_i(\alpha_j T, \overline{x}) : i \in [\ell], j \in [\mathrm{poly}(\ell, k, \deg(P))]\right)$*, with each* α_j *being an arbitrary element of* $\mathbb{F}$*.*

5.8.3 Building up to the proof of Theorem 5.5.1

Theorem 5.8.13 is our main technical theorem building up to Theorem 5.5.1. It shows that for the kind of circuits that show up when testing the divisibility of candidate factors, if composing such a circuit, say C, with the KI generator – instantiated with a polynomial g and a design $\mathcal{S}$ – breaks the nonzeroness of C, then the polynomial g must be "easy".

Theorem 5.8.13. *Let $k > 1$ be any natural number and for every $i \in [\ell]$, let $A_i(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ and $\Phi_i(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ be polynomials satisfying the following properties.*

- *Each $A_i(T, \overline{x}, z)$ is computable by a depth Δ circuit of size s*
- *Each $A_i(T, \overline{x}, z)$ is T-regularized and is monic in z.*
- *For every $i \in [\ell]$, $\Phi_i(T, \overline{x})$ is a truncated, non-degenerate, approximate z-root of $A_i(T, \overline{x}, z)$ of order k.*

Let $\mathcal{S}$ be a (n, σ, μ, ρ)-design and let g be a σ-variate polynomial of degree d. For a new tuple $\overline{w}$ of μ variables distinct from $T, \overline{x}, z$, let us define the polynomial map $\mathbf{KI}_{g,\mathcal{S}}$ using Definition 5.3.9. Then the following is true.

For any circuit C of depth Δ and size s such that the polynomial

$$C'(T, \overline{x}) := C(T, \overline{x}, \Phi_1, \Phi_2, \ldots, \Phi_\ell)$$

is non-zero, if $C'(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}))$ is identically zero, then g can be computed by an algebraic circuit of depth $O(\Delta)$ and size

$$\mathrm{poly}(s, \deg(C'), k) \cdot (\rho \log k)^{\mathrm{poly}(d)}.$$

Proof. The proof proceeds along the lines of the proof of Theorem 5.7.3. In particular, we again apply the hybrid argument to the sequence of substitutions for x_i by g_i and arrive at the point j where the substituted polynomial becomes identically zero for the first time. We again rename the variable x_j as y and set any remaining $\overline{x}$ variables and $\overline{w}$ variables outside the set S_j to field constants while preserving the non-zeroness of the polynomial before the jth substitution. We use $\hat{C}$ to denote the result of applying the sequence of substitutions to the circuit C. We use $\hat{C}'$ to denote the result of applying the sequence of substitutions to the circuit C'. Clearly, it is of the form

$$\begin{aligned}\hat{C}'(T, \overline{w}, y) &= C(T, \hat{g}_1(T, \overline{w}), \ldots, \hat{g}_{j-1}(T, \overline{w}), y, \overline{b}, \hat{\Phi}_1, \ldots, \hat{\Phi}_\ell) \\ &= \hat{C}(T, \overline{w}, y, \hat{\Phi}_1, \ldots, \hat{\Phi}_\ell),\end{aligned}$$

where each $\hat{\Phi}_i(T, \overline{w}, y)$ is a polynomial in $\mathbb{F}[T, \overline{w}_{S_j}, y]$ that satisfies

$$\hat{\Phi}_i(T, \overline{w}, y) = \Phi_i(T, \hat{g}_1(T, \overline{w}), \ldots, \hat{g}_{j-1}(T, \overline{w}), y, \overline{b})$$

for field constants $\overline{b}$ and each $\hat{g}_i$ is obtained from g_i by setting the $\overline{w}$ variables outside the set S_j to field constants according to $\overline{b}$ and hence depends on only $|S_i \cap S_j| \leqslant \rho$ variables. Thus, each $\hat{g}_i$ has a depth-2 circuit of size at most $\rho^{O(d)}$. We also know that $y - g_j(\overline{w})$ divides $\hat{C}'$. From this, we would like to conclude that g_j has a small constant-depth circuit.

We again follow the proof of Theorem 5.7.3, and reduce to the case that $(y - g_j)$ is a factor of multiplicity one. To this end, we will have to work with an appropriately high enough order derivative of $\hat{C}'$ with respect to y (depending on the multiplicity). From Corollary 5.3.2, we get that this derivative is in the $\mathbb{F}[y]$-span of polynomials of the form $\hat{C}'(T, \overline{w}, \beta_i)$ with $\beta_i \in \mathbb{F}$ and $i \in [\deg_y(\hat{C}')]$, and the y-degree of the weights in this linear combination is at most $\deg_y(\hat{C}')$. Let us denote the derivative by B. Thus, we have that

$$B(T, \overline{w}, y) = \sum_{i=0}^{\deg_y(\hat{C}')} \hat{C}'(T, \overline{w}, \beta_i) \cdot \Lambda_{j,i}(y),$$

for univariate polynomials $\Lambda_{j,i}(y)$ of degree at most $\deg_y(\hat{C}')$, B is non-zero and satisfies

$$B(T, \overline{w}, g_j(\overline{w})) \equiv 0,$$

and

$$\frac{\partial B}{\partial y}(T, \overline{w}, g_j(\overline{w})) \neq 0.$$

By shifting the $\overline{w}$ variables if needed, we can assume without loss of generality that

$$\frac{\partial B}{\partial y}(T, \overline{0}, g_j(\overline{0})) \not\equiv 0.$$

Thus, from Lemma 5.4.8, we get that there is a $\kappa \in \mathbb{F}$ and a polynomial Q of degree at most d on $(d+1)$ variables, that satisfies

$$g_j(\overline{w}) \equiv Q(h_0(\kappa, \overline{w}), h_1(\kappa, \overline{w}), \ldots, h_d(\kappa, \overline{w})) \mod \langle \overline{w} \rangle^{d+1},$$

where for every i,

$$h_i(T, \overline{w}) = \frac{\partial B}{\partial y^i}(T, \overline{w}, g_j(\overline{0})) - \frac{\partial B}{\partial y^i}(T, \overline{0}, g_j(\overline{0})) \text{ trunc } \langle \overline{w} \rangle^{d+1}.$$

Given the degree of Q and the number of variables, we get that it has a depth-2 circuit of size at most $\exp(O(d))$. The next claim shows that each h_i has a constant-depth circuit of small size.

Claim 5.8.14. *Each h_i can be computed by a circuit of depth $O(\Delta)$ and size*

$$s' \leqslant \text{poly}(s, \deg(C'), k) \cdot (\rho \log k)^{\text{poly}(d)}$$

We first use the claim to complete the proof of the lemma, and then discuss the proof of the claim.

We take the circuits for $h_i(T, \overline{w})$ — given by Claim 5.8.14, denoted by $V_i(T, \overline{w})$ — and consider the circuit $Q(V_0(\kappa, \overline{w}), V_1(\kappa, \overline{w}), \ldots, V_d(\kappa, \overline{w}))$. Clearly, it is of depth $O(\Delta)$ and size $s' \leqslant \mathrm{poly}(s, \deg(C'), k) \cdot (\rho \log k)^{\mathrm{poly}(d)}$ and satisfies

$$g_j(\overline{w}) \equiv Q(V_0(\kappa, \overline{w}), V_1(\kappa, \overline{w}), \ldots, V_d(\kappa, \overline{w})) \mod \langle \overline{w} \rangle^{d+1} .$$

We can now apply Corollary 5.3.2 to obtain the constant-depth circuit for $g_j(\overline{w})$ with only an additive increase in depth and a polynomial blow up in the size, thereby giving us the theorem. □

Proof of Claim 5.8.14. From Lemma 5.6.1, we know that for any $\beta \in \mathbb{F}$, the polynomials $\hat{A}_i(T, \overline{w}, \beta)$ and $\hat{\Phi}_i(T, \overline{w}, \beta)$, obtained from A and Φ using the substitutions in the above discussion and setting $y = \beta$, continue to satisfy that properties satisfied by A_i, Φ_i in the hypothesis of the lemma.

Thus, for any field constant β, we can invoke Lemma 5.6.3 for each $\hat{A}_i(T, \overline{w}, \beta)$ and $\hat{\Phi}_i(T, \overline{w}, \beta)$ to get that $(\hat{\Phi}_i(T, \overline{w}, \beta) \text{ trunc } \langle \overline{w} \rangle^{d+1})$ can be computed by a circuit of depth $(\Delta + O(1))$ and size s'. We now plug these circuits into the input gates of the constant-depth circuit $\hat{C}$ to get a circuit of depth at most $O(\Delta)$ and size $O(s')$ that computes the polynomial

$$\hat{C}(T, \overline{w}, \beta, (\hat{\Phi}_1(T, \overline{w}, \beta) \text{ trunc } \langle \overline{w} \rangle^{d+1}), \ldots, (\hat{\Phi}_\ell(T, \overline{w}, \beta) \text{ trunc } \langle \overline{w} \rangle^{d+1})) .$$

Now, applying Corollary 5.3.2 with respect to the $\overline{w}$ variables gives us a circuit of depth $O(\Delta)$ and size $\mathrm{poly}(s')$ that computes $\left(\hat{C}'(T, \overline{w}, \beta) \text{ trunc } \langle \overline{w} \rangle^{d+1}\right)$, since by definition $\hat{C}'(T, \overline{w}, y)$ equals $\hat{C}(T, \overline{w}, y, \hat{\Phi}_1, \ldots, \hat{\Phi}_\ell)$.

Similar to Lemma 5.7.1[8], the claim now follows from the definition of B, Corollary 5.3.2 and the fact that the degree of B in T is at most $\deg(C')$. □

We now combine the machinery in Section 5.8.2 with Theorem 5.8.13 to get the following theorem that is directly used in the proof of Theorem 5.5.1.

Theorem 5.8.15. *Let $n \in \mathbb{N}$ be sufficiently large, and let $\overline{x} = (x_1, \ldots, x_n)$. Let $\mathbb{F}$ be a field of characteristic zero. Let $k > 1$ be any natural number. Let $P(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ and $R_1(T, \overline{x}), R_2(T, \overline{x}), \ldots, R_\ell(T, \overline{x}) \in \mathbb{F}[T, \overline{x}]$ be polynomials with the following properties.*

- *$P(T, \overline{x}, z)$ is computable by a depth Δ circuit of size $s \leqslant \mathrm{poly}(n)$*
- *$P(T, \overline{x}, z)$ is T-regularized and is monic in z.*

8 Refer to the part of the proof where we show that $h_i(T, \overline{w})$ can be computed by constant-depth circuits.

- *For every* $i \in [\ell]$, $R_i(T, \overline{x})$ *is a truncated, non-degenerate, approximate* z*-root of* $P(T, \overline{x}, z)$ *of order* k.

Let $Q(T, \overline{x}, z) := \left(\prod_{i \in [\ell]}(z - R_i(T, \overline{x}))\right) \text{ trunc } T^k$.

Let g *be a* σ*-variate degree* d *polynomial and* $\mathcal{S}$ *be a* (n, σ, μ, ρ)*-design. Let* $\mathbf{KI}_{g,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ *be the polynomial map in Definition 5.3.9 defined using* g *and* $\mathcal{S}$*. Then, the following is true.*

If $Q(T, \overline{x}, z)$ *does not divide* $P(T, \overline{x}, z)$ *but* $Q(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ *divides* $P(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$*, then* g *can be computed by an algebraic circuit of depth* $\Delta' = O(\Delta)$ *and size at most*

$$s' = \text{poly}(s, \ell, k, \deg(P)) \cdot (\rho \log k)^{\text{poly}(d)}.$$

In particular, if any depth-Δ' *circuit for* g *requires size greater than* s'*, then* $Q(T, \overline{x}, z)$ *divides* $P(T, \overline{x}, z)$ *if and only if* $Q(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ *divides* $P(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$

Proof. If $Q(T, \overline{x}, z)$ divides $P(T, \overline{x}, z)$, then $Q(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ certainly divides $P(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$.

Suppose $Q(T, \overline{x}, z)$ does not divide $P(T, \overline{x}, z)$. Corollary 5.8.12 tells us that equivalently, there exists a circuit C on at most $\text{poly}(\ell, k, \deg(P))$ variables, and size and degree at most $\text{poly}(\ell, k, \deg(P))$, that is computable by a depth-$O(1)$ circuit such that

$$C\left(z, P_0(T, \overline{x}), \ldots, P_{\deg_z(P)}(T, \overline{x}), \mathcal{R}\right) \not\equiv 0,$$

where, for every i, $P_i(T, \overline{x})$ is the coefficient of z^i in P when viewing it as a univariate in z with coefficients in the ring $\mathbb{F}[T, \overline{x}]$, and $\mathcal{R} = \left(R_i(\alpha_j T, \overline{x}) : i \in [\ell], j \in [\text{poly}(s, k, \deg(P))]\right)$, with each α_j being an element of $\mathbb{F}$.

If $Q(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ divides $P(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$, then

$$C\left(z, P_0(T, \overline{x}) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \ldots, P_{\deg_z(P)}(T, \overline{x}) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \mathcal{R} \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w})\right) \equiv 0,$$

where $\mathcal{R} \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}) := \{R \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}) : R \in \mathcal{R}\}$. Observe that

$$C\left(z, P_0(T, \overline{x}) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \ldots, P_{\deg_z(P)}(T, \overline{x}) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), \mathcal{R} \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w})\right)$$

is the same as

$$C\left(z, P_0(T, \overline{x}), \ldots, P_{\deg_z(P)}(T, \overline{x}), \mathcal{R}\right) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}).$$

Moreover, Lemma 5.6.1 tells us that each $R_i(\alpha_j T, \overline{x})$ is a truncated, non-degenerate, approximate z-root of $P(\alpha_j T, \overline{x}, z)$ of order k, where $P(\alpha_j T, \overline{x}, z)$ is T-regularized and monic in z.

Thus, we can apply Theorem 5.8.13 to $C\left(z, P_0(T, \overline{x}), \ldots, P_{\deg_z(P)}(T, \overline{x}), \mathcal{R}\right)$ to conclude that g can be

computed by an algebraic circuit of depth $\Delta' = O(\Delta)$ and size s' which is at most

$$s' := \mathrm{poly}(s, \ell, k, \deg(P)) \cdot (\rho \log k)^{\mathrm{poly}(d)}.$$

The contrapositive tells us that if every depth-Δ' circuit for g requires size greater than s' obtained as an upper bound above, then $Q(T, \overline{x}, z)$ divides $P(T, \overline{x}, z)$ if and only if $Q(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ divides $P(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$. □

5.8.4 Proof of Theorem 5.5.1

We are now ready to prove Theorem 5.5.1. We start by recalling some notation that we use.

Let $\mathcal{G} = \{g_m\}_{m \in \mathbb{N}}$ be a family of polynomials such that for every $m \in \mathbb{N}$, $g_m \in \mathbb{F}[x_1, \ldots, x_m]$, $d_m := \deg(g_m) \leqslant O(\log\log(m))$. Further, $\mathcal{G}$ has the property that for any depth $\Delta \in \mathbb{N}$, if $\mathcal{C} = \{C_m\}_{m \in \mathbb{N}}$ is a family of depth-Δ circuits computing $\mathcal{G}$, then $\mathcal{C}$ requires size $m^{d_m^{\exp(-O(\Delta))}}$, which is $m^{\omega(1)}$. Theorem 5.3.10 gives us such a family of explicit low-degree polynomials that are hard for constant-depth circuits.

Theorem 5.5.1 (Irreducibility-preserving variable reduction)**.** *Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$. Let $n \in \mathbb{N}$ and $\overline{x} := (x_1, \ldots, x_n)$. Let $P(T, \overline{x}, z)$ be a nonzero polynomial with the following properties.*

- *$P(T, \overline{x}, z)$ is computable by a size $s \leqslant \mathrm{poly}(n)$ and depth Δ circuit.*
- *$P(T, \overline{x}, z)$ is monic in z and T-regularized, with $\deg(P) = D \leqslant \mathrm{poly}(n)$.*
- *$P(T, \overline{x}, z)$ and $P(0, \overline{x}, z) = P(0, \overline{0}, z)$ are squarefree.*

Let $\sigma = O(n^{\varepsilon})$, $\mu = O(\frac{n^{2\varepsilon}}{\log(n)})$, $\rho = O(\log(n))$, and let $\mathcal{S}$ be an (n, σ, μ, ρ)-design. Let $\mathbf{KI}_{g_\sigma,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ be the polynomial map in Definition 5.3.9 defined using the design $\mathcal{S}$ and the polynomial g_σ from the family of hard polynomials $\mathcal{G}$. Then, for an absolute constant[9] $\beta_{\Delta,\varepsilon} \in \mathbb{N}$ and $n \geqslant \beta_{\Delta,\varepsilon}$, the following is true.

A candidate factor[10] $F(T, \overline{x}, z)$ is an irreducible factor of $P(T, \overline{x}, z)$ if and only if $F(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$ is an irreducible factor of $P(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$.

Proof. Suppose $P(T, \overline{x}, z) = \prod_{i=1}^{D}(z - \varphi_i(T, \overline{x}))$, where each $\varphi_i(T, \overline{x})$ is a power series root from $\mathbb{F}[\overline{x}][\![T]\!]$. Then, by Corollary 5.8.5 for $k = \deg_T(P) + 1$, we have that for any subset $S \subseteq [D]$, $F_S(T, \overline{x}, z) := \prod_{i \in S}(z - \varphi_i(T, \overline{x}))$ is an irreducible factor of $P(T, \overline{x}, z)$ in $\mathbb{F}[T, \overline{x}, z]$ if and only if

9 The precise value of $\beta_{\Delta,\varepsilon}$ depends on the exact hardness of the polynomial in $\mathcal{G}$ and the upper bounds obtained in our proofs.

10 See Definition 5.8.2

- $Q_S(T, \overline{x}, z)$ divides $P(T, \overline{x}, z)$, where $Q_S = F_S$ trunc T^k and
- for every $\emptyset \neq U \subsetneq S$ such that $\prod_{i \in U}(z - \alpha_i)$ has all coefficients in $\mathbb{F}$, we have that $Q_U(T, \overline{x}, z) \in \mathbb{F}[T, \overline{x}, z]$ does not divide $P(T, \overline{x}, z)$

For the given parameters, one can verify that the upper bound s' given in Theorem 5.8.15 is at most $\text{poly}(n)$, whereas g_σ requires size $n^{\omega(1)}$ for any constant-depth circuit[11]. Thus, we can apply Theorem 5.8.15 on both the conditions above, to equivalently state that for any subset $S \subseteq [D]$, $F_S(T, \overline{x}, z) = \prod_{i \in S}(z - \varphi_i(T, \overline{x}))$ is an irreducible polynomial factor of $P(T, \overline{x}, z)$ if and only if

- $Q_S(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$ divides $P(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$, and
- for every $\emptyset \neq U \subsetneq S$ such that $\prod_{i \in U}(z - \alpha_i)$ has all coefficients in $\mathbb{F}$, we have that $Q_U(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$ does not divide $P(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$

Lemma 5.6.1 tells us that $P(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$ will remain a nonzero squarefree[12] polynomial that is monic in z and T-regularized, with $\deg_z P = D$. Furthermore, $P(0, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z) = P(0, \overline{0}, z)$ will be squarefree. Thus, we can apply Corollary 5.8.5 to the above two conditions to get that for any subset $S \subseteq [D]$, $F_S(T, \overline{x}, z) = \prod_{i \in S}(z - \varphi_i(T, \overline{x}))$ is an irreducible polynomial factor of $P(T, \overline{x}, z)$ if and only if

- $\prod_{i \in S}(z - \varphi_i(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}))) \in \mathbb{Q}[T, \overline{w}, z]$, and
- for every $\emptyset \neq U \subsetneq S$ such that $\prod_{i \in U}(z - \alpha_i)$ has all coefficients in $\mathbb{F}$, we have that $\prod_{i \in T}(z - \varphi_i(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}))) \notin \mathbb{Q}[T, \overline{w}, z]$

These conditions are true if and only if $F_S(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z) = \prod_{i \in S}(z - \varphi_i(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w})))$ is an irreducible factor of $P(T, \mathbf{KI}_{g_\sigma, \mathcal{S}}(\overline{w}), z)$; this concludes the theorem. □

5.9 PROOF OF CORRECTNESS OF THE ALGORITHM

We will analyze the correctness of our algorithms in a bottom-up fashion, starting with Algorithm 7 and ending with Algorithm 5, which will prove Theorem 5.5.2.

5.9.1 Analysis of Algorithm 7

Suppose $P(T, \overline{x}, z) = \prod_{i \in [m]} G_i(T, \overline{x}, z)$, where $P(T, \overline{x}, z)$ is monic in z, T-regularized with $\deg_T(P) \leqslant \deg(P) := D$ and $\deg_z(P) = D_z$.

11 We chose n sufficiently large enough in the theorem statement ($n > \beta_{\Delta,\varepsilon}$), which implies that for this particular σ, g_σ requires constant-depth circuits larger than s'.

12 Squarefreeness is equivalent to each root being non-degenerate, which is maintained by Lemma 5.6.1.

P is squarefree and $P(0, \bar{0}, z)$ is squarefree as well. The G_is are the irreducible factors of P.

Let $G(T, \bar{x}, z)$ denote an arbitrary G_i. Suppose $G(0, \bar{0}, z) = \prod_{j \in [r]} H_j(z)$ is the factorisation of $G(0, \bar{0}, z)$ into its irreducible factors. Since $P(0, \bar{0}, z)$ is squarefree, each H_j is distinct. If any of the $H_j(z)$ had degree 1, we could've lifted that root via Newton iteration and proceeded with the rest of the algorithm. We will now describe how we deal with the absence of roots of $G(0, \bar{0}, z)$ in $\mathbb{Q}$ using ideas that are standard in the literature (for instance, see [DSS22, Section 6.2]).

Let $H(z)$ be an arbitrary irreducible factor of $G(0, \bar{0}, z)$ and define the field $\mathbb{K} := \frac{\mathbb{Q}[u]}{H(u)}$ for a new variable u. Thus, we are artificially adding a root of $H(z)$ to our field, and calling it u. Every operation in this field is polynomial addition or multiplication over $\mathbb{Q}$ with the variable u, followed by taking the reminder $\bmod H(u)$.

We can now use Newton iteration (Corollary 5.4.4) to compute a truncated, non-degenerate, approximate z-root $\varphi(T, \bar{x}) \in \mathbb{K}[T, \bar{x}]$ satisfying $G(T, \bar{x}, \varphi(T, \bar{x})) \equiv 0 \bmod \langle T \rangle^{2D \cdot D_z + 1}$ and $\varphi(0, \bar{x}) = u \in \mathbb{K}$. The following lemma, standard in the factorisation literature (see [Bür04, Lemma 3.6]), tells us that G is the unique minimal polynomial of φ with the above constraints. The same lemma was stated and proven in Lemma 3.3.2 with some minor differences in notation. Thus, we skip the proof here.

Lemma 5.9.1. *Let $G(T, \bar{x}, z) \in \mathbb{Q}[T, \bar{x}, z]$ be an irreducible polynomial, monic in z with T-degree at most D and z-degree exactly D_z. Let $\varphi(T, x) \in \mathbb{K}[T, \bar{x}]$ be an approximate root of G of order $2D \cdot D_z + 1$, satisfying $G(T, \bar{x}, \varphi(T, \bar{x})) \equiv 0 \mod \langle T \rangle^{2D \cdot D_z + 1}$.*

Now, suppose $B(T, z) \in \mathbb{Q}(\bar{x})[T, z]$ is monic in z with T-degree at most D and z-degree exactly D_z satisfying:

$$B(T, \varphi(T, \bar{x})) \equiv 0 \mod \langle T \rangle^{2D \cdot D_z + 1}$$

Then $B \equiv G$.

5.9.1.1 Linear system for computing the unique minimal polynomial

Let $\mathcal{G} = \{g_m\}_{m \in \mathbb{N}}$ be a family of polynomials such that for every $m \in \mathbb{N}$, $g_m \in \mathbb{F}[x_1, \ldots, x_m]$, $d_m := \deg(g_m) \leqslant O(\log \log(m))$. Further, $\mathcal{G}$ has the property that for any depth $\Delta \in \mathbb{N}$, if $\mathcal{C} = \{C_m\}_{m \in \mathbb{N}}$ is a family of depth-Δ circuits computing $\mathcal{G}$, then $\mathcal{C}$ requires size $m^{d_m^{\exp(-O(\Delta))}}$, which is $m^{\omega(1)}$. Theorem 5.3.10 gives us such a family of explicit low-degree polynomials that are hard for constant-depth circuits.

The following theorem is a variant of Lemma 3.3.3 for obtaining a circuit that computes the minimal polynomial of an approximate root. The proof is mostly the same; the main difference is that we are now

able to eliminate the division gate that showed up due to the use of Cramer's rule in Lemma 3.3.3.

Theorem 5.9.2 (Small division-free circuit for the minimal polynomial of an approximate root)**.** *Fix any* $\Delta \in \mathbb{N}$ *and* $\varepsilon \in (0, 0.5)$*. For an absolute constant* $\beta_{\Delta,\varepsilon} \in \mathbb{N}$*, let* $n > \beta_{\Delta,\varepsilon}$ *and* $\overline{x} = (x_1, \ldots, x_n)$*. Let* $P(T, \overline{x}, z)$ *be a polynomial with the following properties.*

- P *is* T*-regularized and monic in* z *with total degree* D*.*
- $G(T, \overline{x}, z)$ *is an irreducible factor of* P *with* z*-degree* D_z*.*
- $H(z)$ *is an irreducible factor of* $G(0, \overline{x}, z)$*, and* $\mathbb{K}$ *is the field* $\frac{\mathbb{Q}[u]}{H(u)}$*.*

Then, there is a deterministic algorithm $\mathcal{A}_{\Delta,\varepsilon}$ *(with access to* $H(z)$*) which*

- *takes as input a size* s *circuit computing* $\varphi(T, \overline{x}) \in \mathbb{K}[T, \overline{x}]$*, a truncated, approximate* z*-root of* $P(T, \overline{x}, z)$ *of order* $(2D \cdot D_z + 1)$ *such that* $\varphi(0, \overline{x}) = u$ *is a root of* $G(0, \overline{x}, z)$ *over* $\mathbb{K}$*;*
- *outputs a division-free circuit over* $\mathbb{Q}$ *for* $G(T, \overline{x}, z)$ *with size* $\mathrm{poly}(s, D)$*;*
- *and runs in time* $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$*.*

Proof. We would like to compute a polynomial $B(T, z) \in \mathbb{Q}(\overline{x})[T, z]$ that is monic in z with T-degree at most D and z-degree exactly D_z satisfying:

$$B(T, \varphi(T, \overline{x})) \equiv 0 \mod \langle T \rangle^{2D \cdot D_z + 1}$$

where $\varphi(T, \overline{x}) \in \mathbb{K}[T, \overline{x}]$ is computed by a circuit with constants from $\mathbb{K}$. Lemma 5.9.1 guarantees that such a polynomial B has to be the irreducible factor G.

We can separate B into its bihomogeneous components with respect to T and z as

$$B(T, z) = \sum_{i=0}^{d} \sum_{j=0}^{D} B_{i,j} z^i T^j$$

where each $B_{i,j}$ can take a value in the field $\mathbb{Q}(\overline{x})$. Since B is monic in z, $B_{d,0} = 1$ and $B_{d,j} = 0$ for all $j > 0$. The minimal polynomial satisfies

$$B(T, \varphi(T, \overline{x})) = \sum_{i=0}^{d} \sum_{j=0}^{D} B_{i,j} (\varphi(T, \overline{x}))^i T^j \equiv 0 \mod \langle T \rangle^{2D \cdot D_z + 1}$$

which can equivalently be written as $2D \cdot D_z + 1$ many linear constraints, where each constraint expresses that the coefficient of T^l in $B(T, \varphi(T, \overline{x}))$ is zero, for some $l \in \{0, 1, \ldots, 2D \cdot D_z\}$. Let $\varphi^{(i,j)}$ denote

the coefficient of T^j in φ^i. Thus, we have the following linear system in the variables $B_{i,j}$ for $i \in \{0, \ldots, D\}$ and $j \in \{0, \ldots, D_z\}$.

$$B_{d,0} = 1 \tag{5.9.3}$$

$$\forall j \in \{1, \ldots, D\} : B_{d,j} = 0 \tag{5.9.4}$$

$$\forall l \in \{0, 1, \ldots, 2D \cdot D_z + 1\} : \sum_{i=0}^{d} \sum_{j=0}^{l} B_{i,j} (\varphi)^{(i,l-j)} = 0 \tag{5.9.5}$$

At this point, the coefficients of each constraint come from $\mathbb{K}[\overline{x}]$, which means that we can interpret each constraint as a degree $(\deg(H) - 1)$ polynomial in the variable u, with coefficients from $\mathbb{Q}[\overline{x}]$. Since the minimal polynomial of u is H, it follows that $1, u, \ldots, u^{\deg(H)-1}$ are linearly independent over $\mathbb{Q}$ and, in fact, over[13] $\mathbb{Q}[\overline{x}]$. Thus, for a constraint to be equal to zero, the coefficient of each u^r (for $r \in \{0, 1, \ldots, \deg(H) - 1\}$) must be identically zero in $\mathbb{Q}[\overline{x}]$. To compute the coefficient of u^r in $(\varphi)^{(i,j)}$, we can apply Corollary 5.3.2 and get a circuit of size $\mathrm{poly}(s)$; we will denote this circuit by $(\varphi)^{(i,j,r)}$. Thus, we can express the constraints of the form Eq. (5.9.5) using equivalent constraints of the form:

$$\sum_{i=0}^{d} \sum_{j=0}^{l} B_{i,j} (\varphi)^{(i,l-j,r)} = 0$$

for every $r \in \{0, \ldots, (\deg(H) - 1)D_z\}$ and for every $l \in \{0, 1, \ldots, 2D \cdot D_z + 1\}$. The linear system is now over $\mathbb{Q}[\overline{x}]$, and thus, any solution to the linear system is going to be over $\mathbb{Q}(\overline{x})$.

We can express the linear system as $M_\varphi \vec{B} = \vec{c_\varphi}$ for a matrix M_φ with entries in $\mathbb{Q}[\overline{x}]$, the vector of variables $\vec{B} = (B_{i,j})$ and a vector $\vec{c_\varphi} \in (\mathbb{Q}[\overline{x}])^n$. Since the linear system has a unique solution (Lemma 5.9.1), M_φ has full column rank. Using some basic linear algebra (see Lemma 3.3.5), we can rewrite the linear system as $M_\varphi^T M_\varphi \vec{B} = M_\varphi^T \vec{c_\varphi}$, where $M_\varphi^T M_\varphi$ is an invertible square matrix. Thus, we can express the solution to this linear system as $\vec{B} = (M_\varphi^T M_\varphi)^{-1} (M_\varphi^T \vec{c_\varphi})$. In particular, each $B_{i,j}$ can be expressed as $\frac{N_{i,j}(\overline{x})}{\det(M_\varphi^T M_\varphi)}$, where both the numerator and the denominator have circuits of size $\mathrm{poly}(s, D)$.[14]

At this point, we would like to use Strassen's division elimination (Lemma 5.3.4) to write each $B_{i,j}$ as a circuit without division. Strassen's method requires that we find a point $\vec{\gamma} \in \mathbb{Q}^n$ such that $\det(M_\varphi^T M_\varphi)$ is non-zero at $\vec{\gamma}$, but this seems to require a hitting set for circuits since the naive upper bound that we can give for $\det(M_\varphi^T M_\varphi)$ is a small circuit. We will now show that the variable reduction map from Theorem 5.5.1 also preserves the nonzeroness of $\det(M_\varphi^T M_\varphi)$.

13 If $\sum_i p_i(\overline{x}) u^i \equiv 0$, then $\sum_i p_i(\overline{0}) u^i = 0$.

14 Here, we use the well-known fact that the Determinant can be computed efficiently.

Let $\sigma = O(n^{\varepsilon}), \mu = O(\frac{n^{2\varepsilon}}{\log(n)}), \rho = O(\log(n))$, and let $\mathcal{S}$ be an (n, σ, μ, ρ)-design. Let $\mathbf{KI}_{g_\sigma,\mathcal{S}} : \mathbb{F}^{\mu} \to \mathbb{F}^{n}$ be the polynomial map in Definition 5.3.9 defined using the design $\mathcal{S}$ and the polynomial g_σ from the family of hard polynomials $\mathcal{G}$.

Claim 5.9.6. $\det(M_\varphi^T M_\varphi) \circ \mathbf{KI}_{g,\mathcal{S}}(\overline{w}) \not\equiv 0$

Proof. Recall that $\det(M_\varphi^T M_\varphi)$ is non-zero precisely because the irreducible factor $G(T, \overline{x}, z)$ is the unique solution for the linear system. Lemma 5.6.1 tells us that $\varphi(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}))$ is the unique truncated, non-degenerate, approximate z-root of $P(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$ of order $2D \cdot D_z + 1$, satisfying $\varphi(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})) = u = \varphi(0, \overline{x})$. By Theorem 5.5.1, we now know that $\mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})$ preserves the irreducibility of factors, so $G(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$ is irreducible. Thus, if we had first applied $\mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})$ to the polynomial $P(T, \overline{x}, z)$ and computed the truncated, non-degenerate, approximate root $\varphi(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}))$ of order $2D \cdot D_z + 1$ starting from the same point $u \in \mathbb{K}$, Lemma 5.9.1 tells us that we would've still maintained uniqueness of solution for the linear system that computes a monic minimal polynomial of $\varphi(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}))$ of T-degree at most D and z-degree exactly D_z. Moreover, the new linear system is going to be the same as the old linear system, except each $(\varphi)^{(i,l-j,r)}$ will be replaced by $(\varphi \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}))^{(i,l-j,r)}$; this follows because the transformation $\overline{x} \mapsto \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})$ sends every monomial to a polynomial with the same (T, z)-degree and u-degree[15]; in other words, the matrix $M_{\varphi \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})}$ will be equal to $M_\varphi \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})$. Thus, uniqueness of the linear system's solution implies that $\det(M_\varphi^T M_\varphi)(\overline{x}) \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}) \equiv \det(M^T_{\varphi \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})} M_{\varphi \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})}) \not\equiv 0$. □

Thus, to find a point $\vec{\gamma} \in \mathbb{Q}^n$ satisfying $\det(M_\varphi^T M_\varphi)(\vec{\gamma}) \neq 0$, we can first compose it with $\mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w})$ to get a degree $\mathrm{poly}(n)$ polynomial on $\mu = O(n^{2\varepsilon})$ variables. For this low-variate polynomial, we can use the brute force derandomization of the Polynomial Identity Lemma (Lemma 2.3.10) to find a point $\hat{\vec{\gamma}} \in \mathbb{Q}^{\mu}$ such that $\det(M_\varphi^T M_\varphi) \circ \mathbf{KI}_{g_\sigma,\mathcal{S}}$ is nonzero at $\hat{\vec{\gamma}}$; this takes time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$. Thus, $\vec{\gamma} := \mathbf{KI}_{g_\sigma,\mathcal{S}}(\hat{\vec{\gamma}}) \in \mathbb{Q}^n$ will be a point where $\det(M_\varphi^T M_\varphi)(\vec{\gamma}) \neq 0$. We can then use Lemma 5.3.4 to give a circuit of size $\mathrm{poly}(s, D)$ for each $B_{i,j}$, which implies that $G(T, \overline{x}, z) = B(T, z) = \sum_{i=0}^{d} \sum_{j=0}^{D} B_{i,j} z^i T^j$ has a circuit of size $\mathrm{poly}(s, D)$.

□

Theorem 5.9.7 (Correctness of Algorithm 7)**.** *For any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$, Algorithm 7, for sufficiently large n, satisfies the following.*

- *It takes as input a depth-Δ size-s circuit $C_{\tilde{P}}$ computing a square-free degree-D polynomial $\tilde{P}(T, \overline{x}, z) = \prod_{j \in [m]} \tilde{G}_j(T, \overline{x}, z)$, where the*

15 There can be some cancellations among terms of the same (T, z)-degree and u-degree after the transformation, but it does not affect the shape of the matrix for the linear system.

$\tilde{G}_i$s are the irreducible factors of $\tilde{P}$. Further, $\tilde{P}(T, \overline{x}, z)$ is monic in z, T-regularized with respect to $\overline{x}$ and $C_{\tilde{P}}(0, \overline{x}, z) = C_{\tilde{P}}(0, \overline{0}, z)$ is squarefree.

- *It also takes as input the univariate polynomial $\tilde{G}_j(0, \overline{x}, z) = \tilde{G}_j(0, \overline{0}, z)$ for each $j \in [m]$.*
- *It outputs a list $L = \{C_{\tilde{G}_1}(T, \overline{x}, z), \ldots, C_{\tilde{G}_m}(T, \overline{x}, z)\}$, such that each $C_{\tilde{G}_i}(T, \overline{x}, z)$ is a circuit of size $\mathrm{poly}(s, D)$ and depth $\mathrm{poly}(D)$, which computes the irreducible factor $\tilde{G}_i(T, \overline{x}, z)$ of $\tilde{P}(T, \overline{x}, z)$.*
- *It runs in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$.*

Proof. The algorithm iterates over each $\tilde{G}_j$, so let us focus on one such $\tilde{G}_j$. By Theorem 2.3.27, Line 3 of the algorithm gives the correct factorisation of $\tilde{G}_j(0, \overline{0}, z)$ into its irreducible factors, and it runs in time $\mathrm{poly}(\deg(\tilde{G}_j(0, \overline{0}, z)))$. By Corollary 5.4.4, Line 5 will correctly compute a truncated, non-degenerate, approximate z-root $\varphi(T, \overline{x}) \in \mathbb{K}[T, \overline{x}]$ of $\tilde{G}_j(T, \overline{x}, z)$ of order $2D \cdot D_z + 1$ such that $\varphi(0, \overline{x}) = \varphi(0, \overline{0}) = u \in \mathbb{K}$. Moreover, it runs in time $\mathrm{poly}(s, D)$ and outputs a circuit for $\varphi(T, \overline{x})$ of size $\mathrm{poly}(s, D)$. Finally, by Theorem 5.9.2, Line 6 will correctly compute a circuit $C_{\tilde{G}_j}$ for $\tilde{G}_j$, of size $\mathrm{poly}(s, D)$, in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$. □

5.9.2 Analysis of Algorithm 6

Theorem 5.9.8 (Correctness of Algorithm 6)**.** *For any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$, Algorithm 6, for sufficiently large n, satisfies the following.*

- *The input to the algorithm is a depth-Δ size-s circuit C_P computing the squarefree polynomial $P(\overline{x}) = \prod_{j \in [m]} G_j(\overline{x})$, where the G_js are the irreducible factors of P.*
- *The output of the algorithm is a list $L = \{C_{G_1}(\overline{x}), \ldots, C_{G_m}(\overline{x})\}$, such that each $C_{G_j}(\overline{x})$ is a circuit of size $\mathrm{poly}(s, D)$ and depth $\mathrm{poly}(D)$, which computes the irreducible factor $G_j(\overline{x})$.*
- *The algorithm runs in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$.*

Proof. The coefficient of z^D in $\hat{P}(\overline{x}, z) := P(\overline{x} + (\overline{a} \cdot z))/\delta$ is $\mathrm{Hom}_D[P](\overline{a})/\delta = 1$, thus $\hat{P}$ is monic in z, and $\hat{P}$ retains the squarefreeness of P. Moreover, Corollary 5.3.2 tells us that $\mathrm{Hom}_D[P]$ has a size $\mathrm{poly}(s, D)$ and depth $(\Delta + O(1))$ circuit, so Theorem 2.3.13 outputs $\overline{a}$ satisfying $\mathrm{Hom}_D[P](\overline{a}) \neq 0$, in time $\mathrm{poly}(s, D)^{O(n^{\varepsilon})}$.

Since $\hat{P}$ is monic in z and squarefree, Theorem 5.3.5 along with Lemma 3.3.4 tells us that $\mathrm{Disc}_z(\hat{P})(\overline{x}) \not\equiv 0$. In Line 2, $\overline{b}$ is chosen to ensure that $\mathrm{Disc}_z(\hat{P})(\overline{b}) \neq 0$. For $\tilde{P}(T, \overline{x}, z) := \hat{P}((T \cdot \overline{x}) + \overline{b}, z)$, $\mathrm{Disc}_z(\tilde{P}(0, \overline{x}, z)) = \mathrm{Disc}_z(\tilde{P}(0, \overline{0}, z)) = \mathrm{Disc}_z(\hat{P})(\overline{b}) \neq 0$, which implies that $\tilde{P}(0, \overline{x}, z) \in \mathbb{Q}[z]$ is squarefree. Moreover, Theorem 5.3.6 (along with an application of Corollary 5.3.2 for the complexity of $\frac{\partial \hat{P}}{\partial z}$) tells us that

$\mathrm{Disc}_z(\hat{P}) = \mathrm{Res}_z(\hat{P}, \frac{\partial \hat{P}}{\partial z})$ has a size $\mathrm{poly}(s, D)$ and depth $(\Delta + O(1))$ circuit. Thus, we can again use Theorem 2.3.13 to find $\overline{b}$ in time $\mathrm{poly}(s, D)^{O(n^{\varepsilon})}$.
In Line 3, the map $\mathbf{KI}_{g_\sigma,\mathcal{S}}$ can be constructed in time $\exp(O(n^{\varepsilon}))$ by using Lemma 5.3.8.
In Line 4, $\tilde{P}(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$ is an $O(n^{2\varepsilon})$-variate polynomial. Using Theorem 5.4.9, we can factorise $C_{\tilde{P}}(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$. For each $j \in [m]$, let $\tilde{G}_j(T, \overline{x}, z)$ denote $G_j((T \cdot \overline{x}) + (\overline{a} \cdot z) + \overline{b})$; $\tilde{G}_j$ must be irreducible (this is standard in the literature; for a proof, see [Kum+24, Lemma B.7]).

By Theorem 5.5.1, for each $j \in [m]$, $\tilde{G}_j(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ is an irreducible factor of $\tilde{P}(T, \mathbf{KI}_{g_\sigma,\mathcal{S}}(\overline{w}), z)$. Thus, for $j \in [m]$, the factorisation in Line 4 will compute a circuit $C_{\tilde{G}_j}(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$ that computes $\tilde{G}_j(T, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z)$, satisfying the property that $\tilde{G}_j(0, \mathbf{KI}_{g,\mathcal{S}}(\overline{w}), z) = \tilde{G}_j(0, \overline{0}, z)$.

By Theorem 5.9.7, Algorithm 7 correctly computes the list $L' = \{C_{\tilde{G}_1}(T, \overline{x}, z), \ldots, C_{\tilde{G}_m}(T, \overline{x}, z)\}$ in time $\mathrm{poly}(s, D)^{O(n^{\varepsilon})}$. Line 6 computes $C_{G_j}(\overline{x}) = C_{\tilde{G}_1}(1, \overline{x} - \overline{b}, 0)$ for each $G_j(\overline{x})$ correctly.
Overall, the algorithm takes time $\mathrm{poly}(s, D)^{O(n^{\varepsilon})}$ and outputs circuits of size $\mathrm{poly}(s, D)$ for each irreducible factor $G_j(\overline{x})$ of $P(\overline{x})$. □

5.9.3 Analysis of Algorithm 5

Theorem 5.5.2 (Deterministic subexponential time algorithm for factorisation of constant-depth circuits)**.** *Fix any $\Delta \in \mathbb{N}$ and $\varepsilon \in (0, 0.5)$. There exists an algorithm $\mathcal{A}_{\Delta,\varepsilon}$ which, for all sufficiently large n,*

- *takes as input a polynomial $P(\overline{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$ of degree $D \leqslant \mathrm{poly}(n)$ with a depth-Δ, size $s \leqslant \mathrm{poly}(n)$ circuit;*
- *outputs $\mathrm{poly}(s, D)$-sized circuits for each irreducible factor of P, along with the multiplicity of each such factor; and*
- *runs in time $\mathrm{poly}(s, D)^{O(n^{2\varepsilon})}$.*

Proof. Algorithm 5, instantiated with parameters Δ and ε, is the algorithm $\mathcal{A}_{\Delta,\varepsilon}$ claimed in the theorem statement. Theorem 4.2.2 ensures the correctness of Line 1 so that for each $i \in [r]$, $P_i(\overline{x})$ corresponds to the product of irreducible factors of P that have multiplicity i in the factorisation (and r is the maximum multiplicity of an irreducible factor). The algorithm from Theorem 4.2.2 runs in time $\mathrm{poly}(s, D)$ with oracle access to a PIT algorithm for constant-depth circuits; replacing each oracle access by the algorithm in Theorem 2.3.13 results in a running time of $\mathrm{poly}(s, D)^{O(n^{\varepsilon})}$. For each $P_i(\overline{x})$, Theorem 5.9.8 guarantees that the list L_i output by Algorithm 6 in Line 4 is exactly a list of circuits computing the irreducible factors of P_i; moreover, it runs in time $\mathrm{poly}(s, n^D)^{O(n^{2\varepsilon})}$. Since factors of P_i have multiplicity i in P, Line 5 adds the right multiplicity information along with each

circuit for factors of P_i. Every irreducible factor of $P(\overline{x})$ must occur as an irreducible factor in $P_i(\overline{x})$ for some $i \in [r]$; thus, every irreducible factor along with its multiplicity will be included in the output list. $\square$

Part III

PRESENT: POST-FURSTENBERG

Secrets of the past awake,
Lagrange and Furstenberg spoke:
"Constant-depth can conquer roots"
Thus, factors bow to PIT's might.

6 CLOSURE AND ITS ALGORITHMIC CONSEQUENCES

6.1 OVERVIEW

In the previous chapter, we saw an algorithm that outputs all the factors of a constant-depth circuit in essentially the same time as it takes to deterministically do PIT for constant-depth circuits. The primary structural workhorse of that result is a theorem which shows that a hitting-set generator for constant-depth circuit also preserves the factorisation pattern of constant-depth circuits. This is surprising because the PIT instances that capture the irreducibility of certain factors do not *a priori* have a constant-depth circuit upper bound. Thus, one could argue that the contents of the previous chapter feel like mounting evidence towards a closure result.

In this chapter, we first state the theorem which shows that constant-depth circuits are indeed closed under factoring.

Theorem 1.7.8. *[Closure under factorisation [Bha+25b]] Let $\mathbb{F}$ be a field of characteristic zero, f be a polynomial on n variables of degree d over $\mathbb{F}$ and g be a factor of f. Then, the following are true.*

If f can be computed by an algebraic circuit of size s and depth Δ over $\mathbb{F}$, then g can be computed by an algebraic circuit of size $\mathrm{poly}(s, d, n)$ and depth $\Delta + O(1)$ over $\mathbb{F}$.

If f can be computed by an algebraic formula of size s over $\mathbb{F}$, then g can be computed by an algebraic formula of size $\mathrm{poly}(s, d, n)$ over $\mathbb{F}$.

Remark 6.1.1. *While the statement above is stated for fields of characteristic zero, it is also true for fields of sufficiently large characteristic (depending on the degree d) via the same proof.* ◊

While we do not prove the closure theorem in this thesis, we use a building block of the proof – power-series roots closure – towards the main algorithmic result of this chapter: a subexponential time deterministic algorithm that takes a constant-depth circuit and outputs constant-depth circuits for all its factors.

Theorem 1.7.9. *Let $\mathbb{F}$ be the field of rational numbers. Fix any constant $\Delta \in \mathbb{N}$ and $\varepsilon > 0$. Then, there is a deterministic algorithm $\mathcal{A}_{\Delta,\varepsilon}$ that takes as input a size s depth-Δ circuit for a degree d polynomial $P(\overline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ and outputs circuits of size $\mathrm{poly}(s, d)$ and depth $\Delta + O(1)$ for each irreducible factor $g(\overline{x})$ of $P(\overline{x})$, along with their multiplicities. Moreover, $\mathcal{A}_{\Delta,\varepsilon}$ runs in time $\mathrm{poly}(s, d)^{O(n^{\varepsilon})}$.*

BLACKBOX PIT AND DETERMINISTIC FACTORISATION: A result of Kopparty, Saraf and Shpilka [KSS15] showed that deterministic PIT

algorithms for general circuits (in both the blackbox and the whitebox models) implies a deterministic factorisation algorithm for such circuits. We can now extend this connection in the blackbox setting to models such as constant-depth circuits or formulas. However, at the moment, it is unclear to us if such a conclusion also follows from whitebox derandomization of PIT for these models.

RANDOMIZED ALGORITHMS FOR FACTORISATION: By sampling random points instead of using a deterministic PIT algorithm in the aforementioned reduction, we also obtain an efficient randomized algorithm that takes a polynomial with a small formula / constant-depth circuit as input and outputs small formulas / constant-depth circuits for all the irreducible factors of the input, along with their multiplicities.

6.2 PRELIMINARIES

We recall some standard notation and preliminaries in this section.

NOTATION:

- We use $\mathbb{F}[\![x]\!]$ to denote the ring of formal power series, and $\mathbb{F}(\!(x)\!)$ to refer to the ring of Laurent series with respect to the variable x with coefficients from the field $\mathbb{F}$. We use $\overline{\mathbb{F}}$ to refer to the algebraic closure of the field $\mathbb{F}$.
- For a polynomial $f(x) \in \mathbb{F}[x]$ (or more generally in $\mathbb{F}(\!(x)\!)$) and a monomial x^n, we use $[x^n]\{f\}$ to denote the coefficient of x^n in f. For multivariate polynomials such as $f(x, y)$, we will use $[x^n]\{f\}$ by interpreting $f(x) \in \mathbb{F}[y][x]$ and extracting the coefficient of x^n as a function of y.
- A polynomial map $\mathcal{G} : \mathbb{F}^k \to \mathbb{F}^n$, with the mapping $\bar{z} \mapsto (g_1(\bar{z}), \ldots, g_n(\bar{z}))$, is said to be a *hitting-set generator* for a class $\mathcal{C}$ of polynomials if for every $F \in \mathcal{C}$ we have that $F \circ \mathcal{G} = 0$ implies $F = 0$. Abusing notation, we can also view such a map as a homomorphism $\mathcal{G} : \mathbb{F}[\overline{x}] \to \mathbb{F}[\overline{w}]$ such that for every $F \in \mathcal{C}$ we have that $\mathcal{G}(F) = 0$ implies $F = 0$. The degree of the generator is $\max_i(\deg(g_i))$, which is equivalent to $\max_i \deg(\mathcal{G}(x_i))$ for the homomorphism definition.. The hitting-set generator is said to be *explicit* if $\mathcal{G}$ can be computed efficiently.

6.2.1 Reducing to factorising bivariate polynomials

For many of the factorisation applications, it would be convenient to reduce the problem to a *bivariate* setting. The following definition and subsequent lemmas formalize the precise transformation and their

properties. In what follows, it would be convenient to imagine the set of variables instead as $(\overline{x}, y)$ (that is, calling the last variable as y since it plays a slightly different role). We sometimes abuse notation to refer to a set $\overline{x}$ of variables as $(\overline{x}, y)$ by reusing the same names by artificially introducing a variable.

Definition 6.2.1 (Valid pre-processing maps for factorisation)**.** *Let* $\overline{a}, \overline{b} \in \mathbb{F}^{|\overline{x}|}$ *and let* $\mathbb{K} = \mathbb{F}(\overline{x})$*. The homomorphism* $\Psi_{\overline{a},\overline{b}} : \mathbb{F}[\overline{x}, y] \to \mathbb{K}[t, y]$ *given by*

$$\Psi_{\overline{a},\overline{b}} : x_i \mapsto tx_i + a_i y + b_i, \quad \textit{for all } i,$$
$$\Psi_{\overline{a},\overline{b}} : y \mapsto y$$

is said to be a valid pre-processing *map for a polynomial* $F(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ *if* $G(t, y) = \Psi_{\overline{a},\overline{b}}(F)$ *satisfies the following properties:*

1. *The coefficient of* $y^{\deg(F)}$ *in* G *is nonzero.*
2. *If* $\tilde{G}$ *is the squarefree part of* G*, then* $\tilde{G}(0, y)$ *must be squarefree as well.* ◊

Lemma 6.2.2 (Recovering factors from pre-processed factors)**.** *If* $G(t, y) = \Psi_{\overline{a},\overline{b}}(F(\overline{x}, y))$ *for a valid pre-processing map, then* $G(0, y)$ *is a univariate polynomial in* y *over the base field* $\mathbb{F}$*. Furthermore, the factors of* $G(t, y) \in \mathbb{F}(\overline{x})[t, y]$ *are in one-to-one correspondence with the factors of* $F(\overline{x}, y)$*, with the inverse map* $\Psi^{-1}_{\overline{a},\overline{b}}$ *given by* $x_i \mapsto x_i - a_i y - b_i$ *and* $t \mapsto 1$*.*

Proof. For any choice of $\overline{a}, \overline{b}$, the map $\psi_{\overline{a},\overline{b}} : x_i \mapsto x_i + a_i y + b_i$ is an automorphism of the polynomial ring $\mathbb{F}[\overline{x}, y]$ that keeps non-constant polynomials non-constant. Therefore, for any polynomial $F(\overline{x}, y)$, the factors of $F(\overline{x}, y)$ are in one-to-one correspondence with the factors of $\psi_{\overline{a},\overline{b}}(F(\overline{x}, y))$.

If $\Psi_{\overline{a},\overline{b}}$ is a valid pre-processing map for $F(\overline{x}, y)$ and $G(t, y) = \Psi_{\overline{a},\overline{b}}(F) \in \mathbb{F}[\overline{x}][t, y]$ then by Definition 6.2.1 item 62 we have that $G(t, y)$ is monic in y.[1] By Gauss' lemma, $G(t, y)$ is reducible over $\mathbb{F}(\overline{x})$ if and only if $G(t, y)$ is reducible over $\mathbb{F}[\overline{x}]$. If $G(t, y) = G_1(t, y) \cdot G_2(t, y)$ is a non-trivial factorisation, we get a non-trivial factorisation of $\psi_{\overline{a},\overline{b}}(F(\overline{x}, y))$ by setting $t = 1$ since $G_i(t, y)$ is monic for $i = 1, 2$ and will remain non-trivial under the substitution $t = 1$.

Thus, the factors of $G(t, y)$ are in one-to-one correspondence with the factors of $F(\overline{x}, y)$, and we can easily obtain one from the other. □

Lemma 6.2.3. *Let* $F(\overline{x}, y)$ *be a nonzero polynomial of degree* d*. Suppose* $\overline{a}, \overline{b} \in \mathbb{F}^n$ *(where* $n = |\overline{x}|$*) satisfy the following properties:*

1. $\mathrm{Hom}_d(F)(\overline{a}) \neq 0$,
2. $\mathrm{Disc}_y(\tilde{F}(a_1 y + b_1, \ldots, a_n y + b_n, y)) \neq 0$ *where* $\tilde{F}$ *is the squarefree part of* F *and* $\mathrm{Disc}_y(F)$ *denotes the discriminant of* F *with respect to* y*.*

1 Here we just mean that the coefficient lies in $\mathbb{F}$.

Then, $\Psi_{\overline{a},\overline{b}}$ is a valid pre-processing map for $F(\overline{x}, y)$.

Proof. It is easy to observe that the coefficient of y^d in $G(t, y) = \Psi_{\overline{a},\overline{b}}(F)$ is precisely $\mathrm{Hom}_d(F)(\overline{a})$. Thus, this implies Definition 6.2.1 item 62.

By Lemma 6.2.2, if $\tilde{G}$ is the squarefree part of G, then $\tilde{G} = \Psi_{\overline{a},\overline{b}}(\tilde{F})$. Therefore, $\tilde{G}(0, y)$ will be squarefree if and only if $\tilde{F}(a_1 y + b_1, \ldots, a_n y + b_n, y)$ is square free. Hence, $\mathrm{Disc}_y(\tilde{F}(a_1 y + b_1, \ldots, a_n y + b_n, y)) \neq 0$ implies Definition 6.2.1 item 63. □

Remark 6.2.4. *A consequence of the above lemma, along with the Polynomial Identity Lemma is that, for any polynomial $F(\overline{x}, y)$, the map $\Psi_{\overline{a},\overline{b}}$ when $\overline{a}, \overline{b}$ is picked at random is a valid pre-processing map. In particular, valid pre-processing maps always exist.* ◊

6.2.2 Factorisation of monic polynomials into power series

The following are some standard facts about power series roots of polynomials under modest conditions.

Lemma 6.2.5 (Factorisation into power series)**.** *Let $f(t, y) \in \mathbb{K}[t, y]$ be a polynomial that is monic in y such that $f(0, y)$ is squarefree. For each $\alpha \in \overline{\mathbb{K}}$ (the algebraic closure) such that $f(0, \alpha) = 0$, there is a unique power series $\varphi_\alpha(t) \in \overline{\mathbb{K}}[\![t]\!]$ satisfying $\varphi_\alpha(0) = \alpha$ such that $f(t, \varphi_\alpha(t)) = 0$.*
In fact, the polynomial $f(t, y)$ factorises in $\overline{\mathbb{K}}[\![t]\!][y]$ as

$$f(t, y) = \prod_{\alpha \in Z} (y - \varphi_\alpha(t))$$

where Z is the set of roots of $f(0, y)$ in $\overline{\mathbb{K}}$.

The above lemma is essentially folklore and [DSS22, Section 3] gives a formal proof of the above.

6.2.3 PIT for constant-depth circuits

We make use of known hitting-set generators for polynomial size constant-depth circuits. While we can employ the hitting-set generator from Theorem 2.3.13, the statement below is an alternate generator construction from a result of Andrews and Forbes [AF22].

Theorem 6.2.6 (Explicit hitting sets for constant-depth circuits (Theorem 6.8 in the full version of [AF22]))**.** *Let $\mathbb{F}$ be a field of characteristic zero. For every $k \in \mathbb{N}$, there is a hitting-set generator $\mathcal{G}_k : \mathbb{F}[\overline{x}] \to \mathbb{F}[\overline{w}]$ with $|\overline{w}| = n^{1/2^k + o(1)}$ and for the class of $\mathrm{poly}(n)$-size depth $\Delta \leqslant o(\log\log\log n)$ circuits.* □

6.3 CLOSURE UNDER TAKING FACTORS

In this section, we state (without proof) the main closure results.

6.3.1 Complexity of power series roots

Theorem 6.3.1 (Power series roots without multiplicity [Bha+25b]). *Let* $P(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ *be a polynomial computed by a circuit* C*, and let* $\varphi(\overline{x}) \in \mathbb{F}[\![\overline{x}]\!]$ *be a power series satisfying* $\varphi(\overline{0}) = 0$*,* $P(\overline{x}, \varphi(\overline{x})) = 0$ *and* $\partial_y P(\overline{0}, 0) \neq 0$*. Then, for any* $d \in \mathbb{N}$*, there is a circuit* C' *computing* $\mathrm{Hom}_{\leqslant d}[\varphi]$ *such that*

$$\mathrm{size}(C') \leqslant \mathrm{poly}(d, \mathrm{size}(C))$$

$$\mathrm{depth}(C') \leqslant \mathrm{depth}(C) + O(1)$$

6.3.2 Complexity of general factors

We now proceed to state the closure result for general factors. If $f(t, y)$ is a polynomial that is monic in y and is divisible by $g(t, y)$, then over the algebraic closure $\overline{\mathbb{F}}$ we can express $g(0, y) = \prod_{i=1}^{\ell}(y - \alpha_i)$. Thus, a simple way to obtain a constant-depth circuit for $g(t, y)$ *over the algebraic closure* $\overline{\mathbb{F}}$ is to observe that

$$g(t, y) = \mathrm{Hom}_{\leqslant d}\left(\prod_{i=1}^{\ell}(y - C_{\alpha_i}(t))\right)$$

where $C_{\alpha_i}(t)$ is the root of the power series lifted from α_i obtained via Theorem 6.3.1. (We need a few additional ingredients, such as reducing to the square-free case and taking a suitable shift.)

To obtain a circuit for the factors over the base field, we require a few more modifications (and obtaining a circuit over the base field would also be essential for computing the factors algorithmically). We would require the following statement from the recent work of Andrews and Wigderson [AW24] to prove our closure result for general factors.

Theorem 6.3.2 (Theorem I.8 of [AW24]). *Let* $\mathbb{F}$ *be a field of zero or large characteristic. Suppose* $f, g, h \in \mathbb{F}[z]$ *with* $\deg(f), \deg(g), \deg(h) \leqslant d$*. Suppose* $\alpha_1, \ldots, \alpha_d \in \overline{\mathbb{F}}$ *are the roots of* $f(z)$ *with multiplicity, with* $h(\alpha_i) \neq 0$ *for all* i*. Then, for any* $r \in [d]$*,* $\mathbf{Esym}_r(\frac{g(\alpha_1)}{h(\alpha_1)}, \ldots, \frac{g(\alpha_d)}{h(\alpha_d)})$ *can be computed by a circuit of size* $\mathrm{poly}(d)$ *and depth* $O(1)$ *over the coefficients of* f*,* g *and* h*.*

It is worth stressing that the above circuit is over the base field $\mathbb{F}$, and *does not* take the α_i's as input; the inputs are just the coefficients of f, g and h which come from the base field. Moreover, the theorem is algorithmic i.e. given circuits for f, g and h, the circuits for each

$\mathbf{Esym}_r(\frac{g(\alpha_1)}{h(\alpha_1)}, \ldots, \frac{g(\alpha_d)}{h(\alpha_d)})$ can be computed efficiently. For a detailed proof of the closure theorem, we refer to [Bha+25b].

Theorem 6.3.3 (General factors of algebraic circuits [Bha+25b])**.** *Let $\mathbb{F}$ be a field of zero or large enough characteristic, and let $P(\overline{x}) \in \mathbb{F}[\overline{x}]$ be a polynomial on n variables of degree d computed by a circuit C of size s and depth Δ. Then, any factor $g(\overline{x}) \in \mathbb{F}[\overline{x}]$ of P is computable by a circuit of size* $\mathrm{poly}(s, d, n)$ *and depth $\Delta + O(1)$ over $\mathbb{F}$.*

6.4 DETERMINISTIC ALGORITHMS FOR FACTORISATION

As aforementioned, our closure results lead to a clean proof of correctness for the results of Chapter 5, which gave deterministic subexponential time algorithms to output efficient circuits (of potentially unbounded depth) for each of the factors of constant-depth circuit. Moreover, we can output constant-depth circuits for each of the factors.

The core of the algorithm is the following lemma whose proof we will defer to the end of the section. A version of this lemma also appeared in [Bha+25c] but only for a specifically chosen hitting-set generator. The statement below is more general, and the proof is much cleaner.

Lemma 6.4.1 (Irreducibility preservation)**.** *Let $\mathbb{F}$ be a field of zero (or large enough) characteristic. There exists a universal constant $\delta \in \mathbb{N}$ such that the following is true. Let $F(\overline{x}, t, y) \in \mathbb{F}[\overline{x}, t, y]$ be a nonzero degree d polynomial that is computable by a circuit of size s and depth Δ. Suppose F is monic in y, with the property that $F(\overline{x}, 0, y) = F(\overline{0}, 0, y) \in \mathbb{F}[y]$ (i.e., every monomial divisible by an x_i is also divisible by t), and $F(\overline{0}, 0, y)$ is squarefree.*

Let $\mathcal{G} : \mathbb{F}[\overline{x}] \to \mathbb{F}[\overline{w}]$ be a hitting-set generator for the class of size $\mathrm{poly}(s, d)$*, depth $\Delta + \delta$ circuits.*

Then, for every irreducible factor $G(\overline{x}, t, y)$ of $F(\overline{x}, t, y)$ we have that $G \circ \mathcal{G} \in \mathbb{F}[\overline{w}, t, y]$ is also irreducible.

We now proceed with the main theorem of this section.

Theorem 6.4.2. *Fix any constant $\Delta \in \mathbb{N}$ and $\varepsilon > 0$. Then, there is a deterministic algorithm $\mathcal{A}_{\Delta,\varepsilon}$ that takes as input a size s depth-Δ circuit for a degree d polynomial $P(\overline{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$ and outputs circuits of size* $\mathrm{poly}(s, d)$ *and depth $\Delta + O(1)$ for each irreducible factor $g(\overline{x})$ of $P(\overline{x})$, along with their multiplicities. Moreover, $\mathcal{A}_{\Delta,\varepsilon}$ runs in time* $\mathrm{poly}(s, d)^{O(n^{\varepsilon})}$.

Remark 6.4.3. *The theorem continues to be true over any field of zero or sufficiently large characteristic assuming that we have an efficient deterministic algorithm to factor univariates over this field.* ◊

Proof of Theorem 6.4.2. Let $P(\overline{x})$ be an n-variate degree d polynomial given by a circuit of size s and depth Δ. We may assume without loss of generality that $P(\overline{x})$ is squarefree (as the squarefree component[2] can be extracted using Theorem 4.2.2). We outline the rough steps of the algorithm $\mathcal{A}_{\Delta,\varepsilon}$ below and elaborate on the correctness.

1. **(Pre-processing)** Build a circuit C of size $\mathrm{poly}(s)$ and depth $\Delta + O(1)$ for $F(\overline{x}, t, y) = \Psi_{\overline{a},\overline{b}}(P) \in \mathbb{F}[\overline{x}][t, y]$ where $\Psi_{\overline{a},\overline{b}}$ is a valid pre-processing map for $P(\overline{x})$.

2. **(Variable reduction)** For a generator $\mathcal{G} : \mathbb{F}[\overline{x}] \to \mathbb{F}[\overline{w}]$ for size $\mathrm{poly}(s)$, depth $\Delta + O(1)$ circuits, define the polynomial $\tilde{F}(\overline{w}, t, y) := F(\overline{x}, t, y) \circ \mathcal{G}$. (Instantiating with the generator in Theorem 6.2.6, we may assume $|\overline{w}| = n^{\varepsilon}$)

3. **(Factorising variable-reduced polynomial)** Factorise $\tilde{F}(\overline{w}, t, y)$ into irreducibles as $\tilde{F}(\overline{w}, t, y) = \tilde{G_1}(\overline{w}, t, y) \cdots \tilde{G_k}(\overline{w}, t, y)$. Use interpolation to compute the coefficients of $g_i(y) = \tilde{G_i}(\overline{w}, 0, y) = \tilde{G_i}(\overline{0}, 0, y)$ for all $i \in [k]$.

4. **(Building the factors)** For each $j \in [r]$, if $S_j \subset \overline{\mathbb{F}}$ are the roots of $g_j(y)$ in the algebraic closure, define the polynomial

$$G_j(\overline{x}, t, y) := \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in S_j}(y - R_F(\alpha_i))\right)$$

where we define the rational function $R_F(z) \in \mathbb{F}[\overline{x}, t](z)$ as follows

$$R_F(z) = z + \sum_{m=1}^{2d+2}\left(\frac{1}{\partial_y F(0,z)}\right)^{m+1} \cdot [y^{m-1}]\{\partial_y F(t, y+z) \cdot (y \cdot \partial_y F(0,z) - F(t, y+z))^m\}.$$

The rational function $R_F(z)$ arises in the proof of Theorem 6.3.1 (via Furstenberg's identity for implicitly-defined power series), with the property that for any root $\alpha \in \overline{F}$ of $F(\overline{0}, 0, y)$, $R_F(\alpha)$ is equal, modulo high degree terms, to the unique power series root $\varphi(\overline{x}, t) \in \mathbb{F}[\![\overline{x}, t]\!]$ satisfying $\varphi(\overline{0}, 0) = \alpha$ and $F(\overline{x}, t, \varphi(\overline{x}, t)) \equiv 0$ (see [Bha+25b, Corollary 3.3, Theorem 4.3]).

For any i, the coefficient of y^i in $G_j(\overline{x}, t, y)$ is[3] an elementary symmetric polynomial of the evaluations of the rational function

2 In fact, the algorithm from Theorem 4.2.2 outputs the *squarefree decomposition* of the polynomial. The squarefree decomposition of a polynomial $P(\overline{x})$ is a sequence of polynomials $(P_1, \ldots, P_r)$ such that each P_i is a product of exactly those irreducible factors of P that have multiplicity i in its factorisation. In particular, Theorem 4.2.2 immediately gives us the multiplicity of each factor that we obtain from the rest of the algorithm. Since our candidate factors from the algorithm have constant-depth circuits, we can also run a divisibility test on powers of each candidate factor to compute their multiplicities.

3 This is slightly inaccurate, since to get to the coefficients of G_j, we also need to apply $\mathrm{Hom}_{\leqslant d}$.

$R_F(z)$ over the roots of $g_j(y)$. Thus, we can use Theorem 6.3.2 to compute the coefficients of $G_j(\overline{x}, t, y)$ as elements of $\mathbb{F}[\overline{x}, t]$ (i.e. purely over the base field) via $\text{poly}(s)$ size depth $\Delta + O(1)$ circuits.

5. **(Undo pre-processing and return)** Return $\left\{\Psi^{-1}_{\overline{a},\overline{b}}(G_j) \ : \ j \in [k]\right\}$.

We will justify correctness for each of the above steps.

PRE-PROCESSING: By standard interpolation, the highest degree homogeneous part of $P(\overline{x})$ is also computable by size $\text{poly}(s)$, depth $\Delta + O(1)$ sized circuits, and by [AW24] we have that $\text{Disc}_y(P)$ is computable by a size $\text{poly}(s)$, depth $\Delta + O(1)$ circuit. Thus, by Lemma 6.2.3, any hitting set for size $\text{poly}(s)$, depth $\Delta + O(1)$ circuits may be used to compute a valid pre-procesing map $\Psi_{\overline{a},\overline{b}}$.

For what follows, let $F(\overline{x}, t, y) = \Psi_{\overline{a},\overline{b}}(P)$.

VARIABLE REDUCTION: Note that the polynomial $F(\overline{x}, t, y)$ satisfy the requirements of Lemma 6.4.1. Thus, by Lemma 6.4.1, we have that $\mathcal{G}$ preserves the irreducibility of the irreducible factors of $F(\overline{x}, t, y)$.

FACTORISING VARIABLE-REDUCED POLYNOMIAL: Once we have a variable reduced polynomial, any off-the-shelf factorisation algorithm (such as [Lec07]) may be employed to factorise $F \circ \mathcal{G}$ in time $\text{poly}((sd)^{|\overline{w}|})$. Computing the coefficients of $g_k(y)$ can be done via interpolation.

BUILDING THE FACTORS: Let $\tilde{F}(\overline{w}, t, y) = F \circ \mathcal{G} = \tilde{G_1} \cdots \tilde{G_k}$. By Lemma 6.4.1, we have that $F = G_1 \cdots G_k$ is the decomposition of F into irreducibles with $\tilde{G_j}(\overline{w}, t, y) = G_j \circ \mathcal{G}$. Consider an arbitrary irreducible factor $G_j(\overline{x}, t, y)$ of F with coefficients over the field $\mathbb{F}$. As stated in step 4 of the algorithm's outline (and formalized in Lemma 6.4.4), each $G_j(\overline{x}, t, y)$ can be expressed as $\text{Hom}_{\leqslant d}\left(\prod_{i \in U}(y - R_F(\alpha_i))\right)$ for an appropriate set $U \subseteq [r]$. Since the generator is only applied to the $\overline{x}$ variables, we have that $\tilde{G_j}(\overline{w}, 0, y) = G_j(\overline{x} \circ \mathcal{G}, 0, y) = G_j(\overline{0}, 0, y) = \prod_{i \in U}(y - \alpha_i)$. Since we have already computed $\tilde{G_j}$, we have the coefficients of $g_j(y) = G_j(\overline{x}, 0, y) = G_j(\overline{0}, 0, y)$ which are the elementary symmetric polynomials of $\{\alpha_i \ : \ i \in U\}$.

The expression for $G_j(\overline{x}, t, y)$ is symmetric in the roots of $g_j(y)$; in particular, for any i, the coefficient of y^i in G_j is[4] an elementary symmetric polynomial of the evaluations of the rational function $R_F(z)$ over the roots of $g_j(y)$.

We can now use an effective version of the Fundamental Theorem of Symmetric Polynomials, via Theorem 6.3.2, to compute the coefficients

4 This is slightly inaccurate, since to get to the coefficients of G_j, we also need to apply $\text{Hom}_{\leqslant d}$.

of $G_j(\overline{x}, t, y)$ (using the coefficients of $g_j(y)$ instead of the roots of $g_j(y)$) as elements of $\mathbb{F}[\overline{x}, t]$ (i.e. purely over the base field) via poly(s) size depth $\Delta + O(1)$ circuits.

UNDO PRE-PROCESSING: Now that we have obtained the irreducible factors of $F(\overline{x}, t, y) = \Psi_{\overline{a},\overline{b}}(P(\overline{x}))$, Lemma 6.2.2 provides the inverse transformation to obtain the corresponding factors of $P(\overline{x})$.

RUNNING TIME: Finding the right pre-processing map $\Psi_{\overline{a},\overline{b}}$ using $\mathcal{G}$ takes time poly$(s, d)^{O(n^{\varepsilon})}$. The factorisation of the variable-reduced polynomial also runs in time poly$(s, d)^{O(n^{\varepsilon})}$. Rest of the steps take time poly(s, d). Thus, the total running time is poly$(s, d)^{O(n^{\varepsilon})}$.

This completes the proof correctness of Theorem 6.4.2 modulo the proof of Lemma 6.4.1. □

6.4.1 Proof of Lemma 6.4.1

At the core of the algorithm of Bhattacharjee et al [Bha+25c] was a method to characterize variable reductions that preserve the factorisation structure of the polynomial $F(\overline{x}, t, y)$. Recall that $F(\overline{x}, t, y)$ is monic in y and $F(\overline{0}, 0, y)$ is a squarefree.

Lemma 6.4.4 (Lemma 8.3 in [Bha+25c]). *Let $\{\alpha_1, \ldots, \alpha_r\}$ be the roots of $F(\overline{0}, 0, y)$ in $\overline{\mathbb{F}}$. For a subset $S \subset [r]$, define*

$$Q_S(\overline{x}, t, y) = \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in S}(y - R_F(\alpha_i))\right)$$

Then the set of factors of $F(\overline{x}, t, y)$ over the field $\mathbb{F}$ is exactly the same as

$$\mathcal{F} = \left\{ Q_U(\overline{x}, t, y) \ : \ \begin{array}{l} U \subseteq [r] \text{ where } Q_U(\overline{x}, t, y) \in \overline{\mathbb{F}}[\overline{x}, t, y] \text{ divides } F \text{ and} \\ Q_U(\overline{x}, 0, y) = Q_U(\overline{0}, 0, y) = \prod_{i \in U}(y - \alpha_i) \in \mathbb{F}[y] \end{array} \right\}.$$

□

Let $G(\overline{x}, t, y) \in \mathbb{F}[\overline{x}, t, y]$ be an arbitrary irreducible factor of $F(\overline{x}, t, y)$. By the above lemma, there exists some $U \subseteq [d]$ such that $G(\overline{x}, t, y) = Q_U(\overline{x}, t, y)$ with $Q_U(\overline{0}, 0, y)$ having coefficients in $\mathbb{F}$. Note that, since $F(\overline{0}, 0, y)$ is squarefree, distinct elements of $\mathcal{F}$ have distinct set of roots when $\overline{x}, t$ are set to zero. Since $G(\overline{x}, t, y)$ is irreducible, the set U is minimal in the sense that for every $\emptyset \neq U' \subsetneq U$, we have that $Q_{U'}(\overline{x}, t, y)$ has coefficients outside $\mathbb{F}$ or does not divide $F(\overline{x}, t, y)$.

For the sake of contradiction, assume that $\tilde{G}(\overline{w}, t, y) := G \circ \mathcal{G}$ is reducible and $h(\overline{w}, t, y) \in \mathbb{F}[\overline{w}, t, y]$ is a non-trivial factor of $\tilde{G}$. Then, $h(\overline{0}, 0, y)$ divides $\tilde{G}(\overline{0}, 0, y) = G(\overline{0}, 0, y)$ and the set of roots of $h(\overline{0}, 0, y)$ in $\overline{\mathbb{F}}$ is $\{\alpha_i \ : \ i \in U'\}$ for some $\emptyset \neq U' \subsetneq U$.

Consider the polynomial $Q_{U'}(\overline{x}, t, y)$. We observe that $h(\overline{0}, 0, y)$ has coefficients in $\mathbb{F}$ and that $Q_{U'}(t, \overline{x}, y)$ is symmetric with respect to the set $\{\alpha_i \ : \ i \in U'\}$ of roots of $h(\overline{0}, 0, y)$, which implies that $Q_{U'}(\overline{x}, t, y)$ has all coefficients in $\mathbb{F}$ as well (similar to the argument for $G_j(\overline{x}, t, y)$ using the Fundamental Theorem of Symmetric Polynomials in the outline of Theorem 6.4.2). Hence, as argued above, $Q_{U'}(\overline{x}, t, y)$ does not divide $F(\overline{x}, t, y)$.

However, applying Lemma 6.4.4 for $\tilde{G}(\overline{w}, t, y)$, we have

$$h(\overline{w}, t, y) = \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in U'} (y - R_{\tilde{G}}(\alpha_i))\right)$$

As $F(\overline{0}, 0, y) = \tilde{F}(\overline{0}, 0, y)$ is squarefree and $\tilde{F}(\overline{0}, 0, \alpha_i) = 0$ for each $\alpha_i \in U'$, by Lemma 6.2.5 there is a unique power series root $\tilde{\varphi}_i(\overline{w}, t)$ for $\tilde{F}$ mod t^{d+1} that satisfies $\tilde{F}(\overline{w}, t, \tilde{\varphi}_i) = 0$ and $\tilde{\varphi}_i(0) = \alpha_i$. Note that both $R_{\tilde{G}}(\alpha_i)$ and $R_F(\alpha_i) \circ \mathcal{G}$ satisfy these properties. Hence, by the uniqueness of the power series modulo t^{d+1}, we have

$$\begin{aligned}
R_F(\alpha_i) \circ \mathcal{G} &= R_{\tilde{G}}(\alpha_i) \bmod t^{d+1} \\
\implies h(\overline{w}, t, y) &= \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in U'} (y - R_{\tilde{G}}(\alpha_i))\right) \\
&= \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in U'} (y - R_F(\alpha_i) \circ \mathcal{G})\right) \\
&= \mathrm{Hom}_{\leqslant d}\left(\prod_{i \in U'} (y - R_F(\alpha_i))\right) \circ \mathcal{G} = Q_{U'} \circ \mathcal{G}.
\end{aligned}$$

Therefore, we have that $Q_{U'}(\overline{x}, t, y) \in \mathbb{F}[\overline{x}, t, y]$ does not divide $F(\overline{x}, t, y)$ but $h(\overline{w}, t, y) = Q_{U'} \circ \mathcal{G} \in \mathbb{F}[\overline{w}, t, y]$ does divide $\tilde{F}(\overline{w}, t, y) = F(\overline{w}, t, y) \circ \mathcal{G}$.

We can now use the constant-depth circuit DivTest from Lemma 5.8.9, which reduces divisibility testing of a pair of polynomials to an appropriate polynomial identity test. Define $C(\overline{x}, t, y) := \mathrm{DivTest}(y, \overline{\mathrm{coeff}}_y(F(\overline{x}, t, y)), \overline{\mathrm{coeff}}_y(Q_{U'}(\overline{x}, t, y)))$ where $\overline{\mathrm{coeff}}_y(F)$ refers to the vector of coefficients when F is interpreted as a univariate in y (with coefficients involving the other variables). By Lemma 5.8.9, we have that $C(\overline{x}, t, y)$ is a nonzero polynomial since $Q_{U'}$ does not divide F but $C \circ \mathcal{G}$ is zero since $Q_{U'} \circ \mathcal{G}$ divides $F \circ \mathcal{G}$. But C is a circuit of size $\mathrm{poly}(s)$ and depth $\Delta + O(1)$ and hence this violates the assumption that $\mathcal{G}$ is a hitting-set generator for this class. Hence, we must have that $G \circ \mathcal{G}$ continues to be irreducible for every irreducible factor $G(\overline{x}, t, y)$ of $F(\overline{x}, t, y)$. □(Lemma 6.4.1)

Remark 6.4.5. *In Chapter 5 (specifically, Theorem 5.5.2), the polynomials* $R_F(\alpha_i)$ *were instead replaced by truncated power series obtained via Newton*

Iteration, and therefore it was not known if the polynomials $Q \in \mathcal{F}$ *are computable by constant-depth circuits. As a consequence, we could not provide a polynomial size constant depth upper bound for the above circuit* C*. Thus, Theorem 5.5.2 involved a fairly delicate argument to show that the [LST25]+[KI04]+[CKS19] generator maintains the nonzeroness of these non-divisibility identity tests that arises from approximate power series roots obtained via Newton Iteration.*

With Theorem 6.3.1, we now can argue that the circuit C *above is indeed a polynomial size constant-depth circuit and hence* any *generator for this class of circuits would preserve the factorisation of* $F(\overline{x}, t, y)$*.* ◊

6.4.2 Deterministic factorisation from hitting-set generators

The algorithm in Theorem 6.4.2 and its analysis via Lemma 6.4.1 proves a more general statement.

Corollary 6.4.6. *(Informal) Let* $\mathbb{F}$ *be a field of characteristic zero or large enough, and let* $\mathcal{C}$ *be a* robust enough *class of circuits that is closed under small sums and products, substitution by sparse polynomials (thereby admitting interpolation). Consider the larger class* $\mathcal{C}'$ *computing polynomials of the form* $F(g_1, \ldots, g_m)$ *where* F *is computable by a* $\mathrm{poly}(m)$*-sized constant-depth circuit, and each* $g_i \in \mathcal{C}$*.*

If we have a blackbox PIT for the class $\mathcal{C}'$ *running in time* $T(n)$*, then we have a deterministic* $T(\mathrm{poly}(n))$*-time algorithm to factorise polynomials from the class* $\mathcal{C}$*.*

This, for natural subclasses of algebraic circuits — such as algebraic formulas, algebraic branching programs, algebraic circuits, etc. — we have a reduction from factorisation to polynomial identity testing. This generalizes the result of Kopparty, Saraf and Shpilka [KSS15] who established this connection for the class of general algebraic circuits. Further, by solving each PIT instance by random sampling (and using Schwartz-Zippel Lemma), we get an efficient randomized algorithm that takes a polynomial from $\mathcal{C}$ as input and outputs circuits in $\mathcal{C}$ for each irreducible factor, where $\mathcal{C}$ is some robust enough class of polynomials.

However, Theorem 6.4.2 appears to require *blackbox* PITs for the class $\mathcal{C}'$, whereas [KSS15] established such connections even in the whitebox setting. It is an intriguing open question if efficient whitebox algorithms for PIT of $\mathcal{C}'$ would imply efficient deterministic factoring algorithms for $\mathcal{C}$.

7 FUTURE DIRECTIONS

In this thesis, we saw a deterministic algorithm for factoring constant-depth circuits with essentially the same time complexity as deterministic PIT. The two most important observations from this journey are the following.

1. Roots, approximate roots and factors of constant-depth circuits have constant-depth circuits.

2. Irreducibility preservation reduces to divisibility preservation, which reduces to nonzeroness preservation a.k.a PIT.

We conclude this thesis with some open problems.

1. **Whitebox PIT to deterministic factorisation:** Kopparty, Saraf and Shpilka [KSS15] showed that efficient algorithms for PIT for the class of general circuits leads to efficient deterministic factorisation of general circuits, and this connection is for both the whitebox and the blackbox setting for PITs. Although Corollary 6.4.6 extends the blackbox connection to other natural subclasses of circuits (such as formulas, branching programs, constant-depth circuits), establishing a similar connection in the whitebox setting remains open.

2. **Factorization over finite fields of small characteristic:** In [Bha+25a], we show a weaker version of our structural and algorithmic results over fields of small characteristic, mirroring Kaltofen's results for factors of circuits over such fields. There are some fundamental challenges to be overcome before we can extend these results to small characteristic fields in complete generality. One of the simplest-to-state open problems in the area of factorisation of algebraic circuits is the following — over a characteristic p field, if a polynomial f^p is an n-variate, degree d polynomial computed by a $\mathrm{poly}(n, d)$-sized circuit, is f also computable by a $\mathrm{poly}(n, d)$-sized circuit? The answer to this question is unknown even for the setting of general algebraic circuits. Thus, over fields of small characteristic, we do not enjoy the structural results that support the algorithmic results over fields whose characteristic is large or zero.
   A lot of our reasoning also uses the properties of derivatives of polynomials, and these properties need not hold over small fields. Thus, for instance, we cannot use the Discriminant to capture the squarefreeness of a polynomial because the derivative

could vanish if the polynomial is a pth power (where p is the characteristic of the underlying field).

3. **Factor conjecture:** Given $f(\overline{x})$ computable by a circuit of size s with degree potentially as large as 2^s, if $g \mid f$ with $\deg(g) = d$, then is g computable by a size $\mathrm{poly}(s, d)$ circuit? The Factor Conjecture states that this is true. Burgisser [Bür04] showed that the conjecture is true for border computation i.e. the factor g can be approximately computed by a circuit of size $\mathrm{poly}(s, d)$. In a related work [Bha+25a], we show a simpler proof of this theorem via our closure results.

4. **Faster factorisation of sparse polynomials:** While our results imply a deterministic subexponential-time algorithm for factoring sparse polynomials, it is an interesting open problem to get a deterministic algorithm with better time complexity. For the special case of computing low-degree factors, we saw a quasipolynomial time algorithm in Chapter 2.

5. **Reducing other randomized algorithms to PIT:** We showed that for most rich enough complexity classes, the derandomization of PIT is sufficient for derandomizing multivariate factorisation. Are there other interesting tasks that can be derandomized using hitting-set generators *for the same complexity class*?

Part IV

ANNEXURES

BIBLIOGRAPHY

[AF22] Robert Andrews and Michael A. Forbes. "Ideals, determinants, and straightening: proving and using lower bounds for polynomial ideals." In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. ACM, 2022, pp. 389–402. DOI: 10.1145/3519935.3520025. URL: https://doi.org/10.1145/3519935.3520025.

[AW24] Robert Andrews and Avi Wigderson. "Constant-Depth Arithmetic Circuits for Linear Algebra Problems." In: *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*. 2024, pp. 2367–2386. DOI: 10.1109/FOCS61266.2024.00138.

[AS03] Sanjeev Arora and Madhu Sudan. "Improved Low-Degree Testing and its Applications." In: *Comb.* 23.3 (2003), pp. 365–426. DOI: 10.1007/s00493-003-0025-0. URL: https://doi.org/10.1007/s00493-003-0025-0.

[Beh+25] Amik Raj Behera, Nutan Limaye, Varun Ramanathan, and Srikanth Srinivasan. "New Bounds for the Ideal Proof System in Positive Characteristic." In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Ed. by Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis. Vol. 334. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 22:1–22:20. ISBN: 978-3-95977-372-0. DOI: 10.4230/LIPIcs.ICALP.2025.22. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2025.22.

[BT88] Michael Ben-Or and Prasoon Tiwari. "A deterministic algorithm for sparse multivariate polynomial interpolation." In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, pp. 301–309.

[Ben+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. "Proximity Gaps for Reed–Solomon Codes." In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 900–909. DOI: 10.1109/FOCS46700.2020.00088.

[Bha+20] Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. "Decoding Multivariate Multiplicity Codes on Product Sets." In: *Electron. Colloquium Comput. Complex.* TR20-179 (2020). ECCC: TR20-179. URL: https://eccc.weizmann.ac.il/report/2020/179.

[BDS25] C. S. Bhargav, Prateek Dwivedi, and Nitin Saxena. *A primer on the closure of algebraic complexity classes under factoring*. 2025. arXiv: 2506.19604 [cs.CC]. URL: https://arxiv.org/abs/2506.19604.

[BSV18] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. "Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree." In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*. Ed. by Mikkel Thorup. IEEE Computer Society, 2018, pp. 485–496. DOI: 10.1109/FOCS.2018.00053. URL: https://doi.org/10.1109/FOCS.2018.00053.

[Bha+25a] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Constant-depth circuits for polynomial GCD over any characteristic*. 2025. arXiv: 2506.23220 [cs.CC]. URL: https://arxiv.org/abs/2506.23220.

[Bha+25b] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu S. Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Closure under factorization from a result of Furstenberg*. 2025. arXiv: 2506.23214 [cs.CC]. URL: https://arxiv.org/abs/2506.23214.

[Bha+25c] Somnath Bhattacharjee, Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Deterministic factorization of constant-depth algebraic circuits in subexponential time*. 2025. arXiv: 2504.08063 [cs.CC]. URL: https://arxiv.org/abs/2504.08063.

[Bog05] Andrej Bogdanov. "Pseudorandom generators for low degree polynomials." In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 21–30. DOI: 10.1145/1060590.1060594. URL: https://doi.org/10.1145/1060590.1060594.

[Bür04] Peter Bürgisser. "The Complexity of Factors of Multivariate Polynomials." In: *Foundations of Computational Mathematics* 4.4 (Sept. 2004), pp. 369–396. ISSN: 1615-3383. DOI: 10.1007/s10208-002-0059-5. URL: http://dx.doi.org/10.1007/s10208-002-0059-5.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A. Shokrollahi. *Algebraic Complexity Theory*. Vol. 315. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1997.

[CKS19] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. "Closure Results for Polynomial Factorization." In: *Theory of Computing* 15.13 (2019), pp. 1–34. DOI: 10.4086/toc.2019.v015a013. URL: https://theoryofcomputing.org/articles/v015a013.

[Cla+91] Michael Clausen, Andreas W. M. Dress, Johannes Grabmeier, and Marek Karpinski. "On Zero-Testing and Interpolation of k-Sparse Multivariate Polynomials Over Finite Fields." In: *Theoretical Computer Science* 84.2 (1991), pp. 151–164.

[DL78] Richard A. DeMillo and Richard J. Lipton. "A Probabilistic Remark on Algebraic Program Testing." In: *Information Processing Letters* 7.4 (1978), pp. 193–195. DOI: 10.1016/0020-0190(78)90067-4. URL: http://dx.doi.org/10.1016/0020-0190(78)90067-4.

[DbT24] Pranjal Dutta, Amit Sinhaba bu, and Thomas Thierauf. *Derandomizing Multivariate Polynomial Factoring for Low-degree factors*. 2024. arXiv: 2411.17330 [cs.CC]. URL: https://arxiv.org/abs/2411.17330.

[DG24] Pranjal Dutta and Sumanta Ghosh. "SIGACT News Complexity Theory Column 121: Advances in Polynomial Identity Testing." In: *SIGACT News* 55.2 (June 2024), pp. 53–88. ISSN: 0163-5700. DOI: 10.1145/3674159.3674165. URL: https://doi.org/10.1145/3674159.3674165.

[DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. "Discovering the Roots: Uniform Closure Results for Algebraic Classes Under Factoring." In: *J. ACM* 69.3 (June 2022). ISSN: 0004-5411. DOI: 10.1145/3510359. URL: https://doi.org/10.1145/3510359.

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. "Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits." In: *SIAM J. Comput.* 39.4 (2009), pp. 1279–1293. DOI: 10.1137/080735850. URL: http://dx.doi.org/10.1137/080735850.

[Ell69] W.J. Ellison. "A 'Waring's Problem' for homogeneous forms." In: *Proceedings of the Cambridge Philosophical Society* 65 (1969), pp. 663–672. DOI: 10.1017/S0305004100003455.

[Fis94] Ismor Fischer. "Sums of like powers of multivariate linear forms." In: *Mathematics Magazine* 67.1 (1994), pp. 59–61. DOI: 10.2307/2690560.

[For15] Michael A. Forbes. "Deterministic Divisibility Testing via Shifted Partial Derivatives." In: *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*. FOCS '15. USA: IEEE Computer Society, 2015, pp. 451–465. ISBN: 9781467381918. DOI: 10.1109/FOCS.2015.35. URL: https://doi.org/10.1109/FOCS.2015.35.

[FS15] Michael A. Forbes and Amir Shpilka. "Complexity Theory Column 88: Challenges in Polynomial Factorization1." In: *SIGACT News* 46.4 (Dec. 2015), pp. 32–49. ISSN: 0163-5700. DOI: 10.1145/2852040.2852051. URL: https://doi.org/10.1145/2852040.2852051.

[Fur67] Harry Furstenberg. "Algebraic functions over finite fields." In: *Journal of Algebra* 7.2 (1967), pp. 271–277.

[GK85] J. von zur Gathen and E. Kaltofen. "Factoring Sparse Multivariate Polynomials." In: *Journal of Computer and System Sciences* 31.2 (1985), pp. 265–287. DOI: 10.1016/0022-0000(85)90044-3. URL: http://dx.doi.org/10.1016/0022-0000(85)90044-3.

[Gat87] Joachim von zur Gathen. "Factoring polynomials and primitive elements for special primes." In: *Theoretical Computer Science* 52 (1987), pp. 77–89.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139856065.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[GK87] Dima Yu Grigoriev and Marek Karpinski. "The matching problem for bipartite graphs with polynomially bounded permanents is in NC." In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. IEEE. 1987, pp. 166–172.

[GKS90] Dima Yu Grigoriev, Marek Karpinski, and Michael F Singer. "Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields." In: *SIAM Journal on Computing* 19.6 (1990), pp. 1059–1063.

[Guo+19] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. "Derandomization from Algebraic Hardness: Treading the Borders." In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Ed. by David Zuckerman. IEEE Computer Society, 2019, pp. 147–157. DOI: 10.1109/FOCS.2019.00018. URL: https://doi.org/10.1109/FOCS.2019.00018.

[HS80] Joos Heintz and Claus-Peter Schnorr. "Testing Polynomials which Are Easy to Compute (Extended Abstract)." In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*. 1980, pp. 262–272. DOI: 10.1145/800141.804674.

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds." In: *Computational Complexity* 13.1-2 (2004). Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pp. 1–46. DOI: 10.1007/s00037-004-0182-6.

[Kal85] Erich Kaltofen. "Polynomial-Time Reductions from Multivariate to Bi- and Univariate Integral Polynomial Factorization." In: *SIAM Journal of Computing* 14.2 (1985), pp. 469–489. DOI: 10.1137/0214035.

[Kal89] Erich Kaltofen. "Factorization of Polynomials Given by Straight-Line Programs." In: *Randomness and Computation*. JAI Press, 1989, pp. 375–412. URL: https://users.cs.duke.edu/~elk27/bibliography/89/Ka89_slpfac.pdf.

[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). Computational algebraic complexity editorial, pp. 301–320. ISSN: 0747-7171. DOI: https://doi.org/10.1016/S0747-7171(08)80015-6. URL: https://www.sciencedirect.com/science/article/pii/S0747717108800156.

[KT88] Erich L. Kaltofen and Barry M. Trager. "Computing with Polynomials Given By Black Boxes for Their Evaluation: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators." In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*. IEEE Computer Society, 1988, pp. 296–305. DOI: 10.1109/SFCS.1988.21946. URL: https://doi.org/10.1109/SFCS.1988.21946.

[KS01] Adam Klivans and Daniel A. Spielman. "Randomness efficient identity testing of multivariate polynomials." In: *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*. 2001, pp. 216–223. DOI: 10.1145/380752.380801.

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. "Equivalence of Polynomial Identity Testing and Polynomial Factorization." In: *Computational Complexity* 24.2

(2015). Preliminary version in the *29th Annual IEEE Conference on Computational Complexity (CCC 2014)*, pp. 295–331. DOI: 10.1007/s00037-015-0102-y. URL: http://dx.doi.org/10.1007/s00037-015-0102-y.

[KY08] Swastik Kopparty and Sergey Yekhanin. "Detecting Rational Points on Hypersurfaces over Finite Fields." In: *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*. 2008, pp. 311–320. DOI: 10.1109/CCC.2008.36.

[KRS23] Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. "Deterministic Algorithms for Low-degree factors of Constant Depth Circuits." In: *CoRR* abs/2309.09701 (2023). DOI: 10.48550/ARXIV.2309.09701. arXiv: 2309.09701. URL: https://doi.org/10.48550/arXiv.2309.09701.

[Kum+24] Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. *Towards Deterministic Algorithms for Constant-Depth Factors of Constant-Depth Circuits*. 2024. arXiv: 2403.01965 [cs.CC]. URL: https://arxiv.org/abs/2403.01965.

[KS19] Mrinal Kumar and Ramprasad Saptharishi. "Hardness-randomness tradeoffs for algebraic computation." In: *Bulletin of EATCS* 3.129 (2019).

[Lec07] Grégoire Lecerf. "Improved dense multivariate polynomial factorization algorithms." In: *Journal of Symbolic Computation* 42.4 (2007), pp. 477–494. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2007.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0747717107000119.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. "Factoring polynomials with rational coefficients." In: *Mathematische Annalen* 261.4 (1982), pp. 515–534. DOI: 10.1007/BF01457454.

[LST25] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. "Superpolynomial lower bounds against low-depth algebraic circuits." In: *Journal of the ACM* 72.4 (2025), pp. 1–35.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs Randomness." In: *Journal of Computer and System Sciences* 49.2 (1994). Available on citeseer:10.1.1.83.8416, pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[NW97] Noam Nisan and Avi Wigderson. "Lower bounds on arithmetic circuits via partial derivatives." In: *Computational Complexity* 6.3 (1997). Available on citeseer:10.1.1.90.2644, pp. 217–234. DOI: 10.1007/BF01294256.

[Oli15] Rafael Oliveira. "Factors of low individual degree polynomials." In: *Proceedings of the 30th Conference on Computational Complexity*. CCC '15. Portland, Oregon: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 198–216. ISBN: 9783939897811.

[Ore22] Øystein Ore. "Über höhere Kongruenzen." In: *Norsk Mat. Forenings Skrifter* 1.7 (1922), p. 15.

[PS94] Alexander Polishchuk and Daniel A. Spielman. "Nearly-Linear Size Holographic Proofs." In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC '94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, pp. 194–203. ISBN: 0897916638. DOI: 10.1145/195058.195132. URL: https://doi.org/10.1145/195058.195132.

[Roc18] Daniel S. Roche. "What Can (and Can't) We Do with Sparse Polynomials?" In: *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*. ISSAC '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 25–30. ISBN: 9781450355506. DOI: 10.1145/3208976.3209027. URL: https://doi.org/10.1145/3208976.3209027.

[Sap14] Ramprasad Saptharishi. "Recent Progress on Arithmetic Circuit Lower Bounds." In: *Bulletin of the EATCS* 114 (2014). URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/293.

[Sch80] Jacob T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities." In: *Journal of the ACM* 27.4 (1980), pp. 701–717. DOI: 10.1145/322217.322225.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Available from http://shoup.net/ntb/. New York: Cambridge University Press, 2009.

[SV10] Amir Shpilka and Ilya Volkovich. "On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors." In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*. Vol. 6198. Lecture Notes in Computer Science. Springer, 2010, pp. 408–419. DOI: 10.1007/978-3-642-14165-2_35. URL: https://doi.org/10.1007/978-3-642-14165-2%5C_35.

[SW01] Amir Shpilka and Avi Wigderson. "Depth-3 arithmetic circuits over fields of characteristic zero." In: *Computational Complexity* 10.1 (2001). Preliminary version in the *14th Annual IEEE Conference on Computational Complexity (CCC 1999)*, pp. 1–27. DOI: 10.1007/PL00001609.

[SY10] Amir Shpilka and Amir Yehudayoff. "Arithmetic Circuits: A survey of recent results and open questions." In: *Foundations and Trends in Theoretical Computer Science* 5 (3–4 Mar. 2010), pp. 207–388. ISSN: 1551-305X. DOI: http://dx.doi.org/10.1561/0400000039.

[ST20] Amit Sinhababu and Thomas Thierauf. "Factorization of Polynomials Given By Arithmetic Branching Programs." In: *35th Computational Complexity Conference (CCC 2020)*. Ed. by Shubhangi Saraf. Vol. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 33:1–33:19. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.33. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12585.

[Str73] Volker Strassen. "Vermeidung von Divisionen." ger. In: *Journal für die reine und angewandte Mathematik* 264 (1973), pp. 184–202. URL: http://eudml.org/doc/151394.

[Sud98] Madhu Sudan. *Lecture notes for the course 'Algebra and Computation'*. Available from http://people.csail.mit.edu/madhu/FT98/. 1998.

[Val79] Leslie G. Valiant. "Completeness Classes in Algebra." In: *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*. 1979, pp. 249–261. DOI: 10.1145/800135.804419.

[Val+83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. "Fast Parallel Computation of Polynomials Using Few Processors." In: *SIAM Journal of Computing* 12.4 (1983). Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*, pp. 641–644. DOI: 10.1137/0212043.

[Vol15] Ilya Volkovich. "Deterministically Factoring Sparse Polynomials into Multilinear Factors and Sums of Univariate Polynomials." In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*. Vol. 40. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 943–958. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2015.943. URL: https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.943.

[Vol17] Ilya Volkovich. "On Some Computations on Sparse Polynomials." In: vol. 81. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 48:1–48:21. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2017.48. URL: https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2017.48.

[Zip79] Richard Zippel. "Probabilistic algorithms for sparse polynomials." In: *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*. Vol. 72. Lecture Notes in Computer Science. Springer, 1979, pp. 216–226. DOI: 10.1007/3-540-09519-5_73. URL: http://dx.doi.org/10.1007/3-540-09519-5_73.

PUBLICATIONS

This thesis includes results from the following papers, based on collaborations with Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Suresh Rai, Ramprasad Saptharishi, Shubhangi Saraf and Ben Lee Volk. Note that we only include the algorithmic results from [Bha+25b]; the structural (closure) results will be a part of Shanthanu Suresh Rai's thesis.

[Bha+25a] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu S. Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Closure under factorization from a result of Furstenberg*. 2025. arXiv: 2506.23214 [cs.CC]. URL: https://arxiv.org/abs/2506.23214.

[Bha+25b] Somnath Bhattacharjee, Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Deterministic factorization of constant-depth algebraic circuits in subexponential time*. 2025. arXiv: 2504.08063 [cs.CC]. URL: https://arxiv.org/abs/2504.08063.

[KRS23] Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. "Deterministic Algorithms for Low-degree factors of Constant Depth Circuits." In: *CoRR* abs/2309.09701 (2023). DOI: 10.48550/ARXIV.2309.09701. arXiv: 2309.09701. URL: https://doi.org/10.48550/arXiv.2309.09701.

[Kum+24] Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. *Towards Deterministic Algorithms for Constant-Depth Factors of Constant-Depth Circuits*. 2024. arXiv: 2403.01965 [cs.CC]. URL: https://arxiv.org/abs/2403.01965.

The following papers were also written during the course of the thesis work but they are not included in this thesis. In [Bha+25a], we explore some interesting applications of the closure result from [Bha+25b], and it will be a part of Shanthanu Suresh Rai's thesis. [Beh+25] contains results in algebraic proof complexity that are thematically different from this thesis and hence was not included in the thesis.

[Beh+25] Amik Raj Behera, Nutan Limaye, Varun Ramanathan, and Srikanth Srinivasan. "New Bounds for the Ideal Proof System in Positive Characteristic." In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Ed. by Keren Censor-Hillel, Fabrizio

Grandoni, Joël Ouaknine, and Gabriele Puppis. Vol. 334. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 22:1–22:20. ISBN: 978-3-95977-372-0. DOI: 10.4230/LIPIcs.ICALP.2025.22. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2025.22.

[Bha+25] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Constant-depth circuits for polynomial GCD over any characteristic*. 2025. arXiv: 2506.23220 [cs.CC]. URL: https://arxiv.org/abs/2506.23220.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/