ON POLYNOMIAL IDENTITY TESTING AND RELATED PROBLEMS

by

Partha Mukhopadhyay

THE INSTITUTE OF MATHEMATICAL SCIENCES, CHENNAI.

A thesis submitted to the
Board of Studies in Mathematical Sciences

In partial fulfillment of the requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

September 2009

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we recommend that the dissertation prepared by **Partha Mukhopadhyay** entitled "On Polynomial Identity Testing and Related Problems" may be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

- **Date :**
Chairman and Convener: V. Arvind

- **Date :**
Member : Manindra Agrawal

- **Date :**
Member : Meena Mahajan

- **Date :**
Member : Venkatesh Raman

- **Date :**
Member : K. V. Subrahmanyam

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

- **Date :**
Guide : V. Arvind

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and the work has not been submitted earlier as a whole or in part for a degree/diploma at this or any other Institution or University.

Partha Mukhopadhyay

To my parents and family members

ACKNOWLEDGEMENTS

Many wonderful people have influenced my personal and professional life, over the years. This acknowledgement is for all of them. I am sure, I won't be able to mention many of them even after next few paragraphs. My sincere apology to all of them.

I am deeply indebted to my advisor V. Arvind. Over the last few years, his guidance was instrumental in all aspects of my life and made my stay in IMSc very much enjoyable. Whatever I have learnt about doing research, I have learnt from him. His presence remains in every part of this thesis and in my research: posing a new research problem, solving problems and showing me how to write technical articles. I am grateful to him for sharing with me, his deep insights in computer science and spending hours for technical discussions. Thank you Arvind for your faith in me. That has given me a great confidence to pursue research.

I am grateful to all the faculty members of IMSc and CMI for their teaching and encouraging discussions. Particularly, my sincere thanks to Meena Mahajan, Venkatesh Raman, C.R Subramanian, Kamal Lodaya, R. Ramanujam, K.V. Subramanyam, and Samir Datta. My overall understanding of theoretical computer science is greatly improved by your teaching and several discussions with you. Meena, thank you once again for organizing those exciting theory seminars and arranging a great party in your house.

I thank Amir Shpilka, Manindra Agrawal, and Ashwin Nayak for their comments and suggestions on some of our results. Special thanks to Bireswar Das and Srikanth Srinivasan for their collaborations. Their clear thinking helped me a lot in doing research. Thanks to Piyush Kurur for many interesting technical discussions during my early days in IMSc.

I would like to thank Madhavan Mukund and S. Kesavan for their help in arranging grants for conference travels. I thank administrative members of IMSc for their help during last few years. I take this opportunity to thank R. Balasubramanian, the director of IMSc, for his help in every academic and non-academic aspect.

My interest in theoretical computer science was initialized during my MTech days in ISI, Kolkata. Thanks to Rana Barua, K. Sikdar, and Guruprasad Kar for their great teaching. Learning "quantum magics" from Guruprasad was always a unique experience.

I am fortunate to have many wonderful friends in IMSc during my stay. I thank all my fellow members in computer science department. Especially, thanks to Bireswar, Somnath, Raghu, Pushkar, Aravind, Jayalal, Srikanth, Nutan, Rahul, Narayanan, Prajakta, Kunal, and Saket for several interesting conversations. Thank you Somnath, for making many dinners memorable. A great thanks to Narayanan and Jayalal for their help in preparing LaTeX template used in this thesis. Thanks to my office mates, Prajakta (it is a family matter now!), Arindam and Pushkar for all the fun moments in the office.

I have found several good friends outside the computer science department as well, who made my stay in IMSc memorable. A few of them include Arijit, Sarbeswar, Gaurav, Vinu, Ved, Amit, Pradeep, Swagata, Pooja, Kamil, Nilanjan, Anisa. Thank you Arijit for your help during tough times.

Finally, and most importantly, I would like to thank my family members for supporting me and my decisions in every aspect. I have no words to express my heartfelt thanks to my parents. Nothing would have been possible without their love and continuous support. Thank you Prajakta, for your care, support and giving me confidence when needed. Thank you Aai, Papa and dear Prachi, you helped me a lot. I love all of you and this thesis is for you.

Abstract

In this thesis we study the polynomial identity testing problem and its connection to several other important complexity-theoretic problems. We consider polynomial identity testing problem over commutative and noncommutative models of computation. We obtain efficient randomized identity testing algorithm over finite rings. We establish connections between ideal membership problem and identity testing and as a byproduct, we get new understanding of identity testing for depth-3 circuits. Over noncommutative model, we give new efficient deterministic identity testing and polynomial interpolation algorithms for small degree and sparse polynomials. Related to identity testing, we obtain interesting derandomization consequences of the Isolation Lemma in the context of circuit size lower bounds. We also obtain a query efficient quantum algorithm for testing if a given polynomial is an identity (i.e zero at all the points) for a given ring. We summarize the main results of the thesis.

Algorithmic results over finite ring:

Given $\epsilon > 0$, we show an ϵ-uniform randomized polynomial-time algorithm to sample from finite black-box rings. This is similar in spirit to Babai's efficient randomized sampling algorithm for finite black-box groups [Bab91]. We obtain a polynomial-time quantum algorithm to compute a basis representation for a black-box ring. Using this result, we give a polynomial-time quantum algorithm to test whether a given black-box ring is a field. Computing the basis representation is based on the sampling algorithm.

Given a polynomial $f(x_1, x_2, \cdots, x_n)$ by an arithmetic circuit C over any finite commutative *ring* R (with unity), we give a randomized polynomial-time algorithm to test if $f \equiv 0$ in $R[x_1, x_2, \cdots, x_n]$. Similar results were known over fields (Schwartz-Zippel Lemma [Sch80, Zip79]) and over $\mathbb{Z}/n\mathbb{Z}$ [AB03] for n composite.

Ideal membership and polynomial identity testing: Given a polynomial f by an arithmetic circuit C, and a monomial ideal I by a set of *constant* number of monomials as generators, we give a randomized polynomial-time algorithm to test if $f \in I$. This algorithm is analogous to the randomized Schwartz-Zippel test for polynomial identity testing. We show that the problem is coNP-hard when the number of generators of the given monomial ideal is not a constant. In that case, we show an upper bound in the counting hierarchy.

Given a monomial ideal I generated by a constant number of monomials, and a polynomial f computed by a depth 3 circuit with bounded top gate fan-in, we give a

deterministic polynomial-time algorithm to test if $f \in I$. The algorithmic ideas are from [KS07], but the correctness proof is new and based on Gröbner basis theory. This approach gives us a different understanding of the identity testing algorithm for such circuits.

Noncommutative polynomial identity testing: We give a deterministic polynomial-time algorithm for identity testing of *noncommutative circuits* computing sparse and small degree polynomials. Before our work, in the noncommutative model, deterministic polynomial-time identity testing algorithm was known for noncommutative arithmetic formulas [RS05]. Our algorithm uses new ideas from automata theory. We give a deterministic polynomial-time interpolation algorithm for a sparse and small degree noncommutative polynomial given by either an arithmetic circuit or by a black-box.

Isolation lemma and polynomial identity testing: We establish connection between derandomization of the isolation lemma and circuit size lower bounds. We formulate a reasonably restricted version of the isolation lemma and show that a subexponential-time (non black-box) derandomization (of this version) implies either NEXP $\not\subset$ P/poly or permanent has no polynomial-size noncommutative arithmetic circuit. Our result can be considered similar in flavour to the result of Impagliazzo-Kabanets [KI03].

We show that a stronger (*black-box*) derandomization of the same version of the isolation lemma means that an explicit polynomial in the commutative model has no subexponential-size arithmetic circuit. The proof idea is based on the results on pseudo-random generators for arithmetic circuits [Agr05].

Query complexity of multilinear identity testing:

Consider any finite ring R (commutative or noncommutative) and any multilinear polynomial $f(x_1, x_2, \cdots, x_m)$ over $R[x_1, x_2, \cdots, x_m]$ or $R\{x_1, x_2, \cdots, x_m\}$ (i.e. either x_i, x_j are commuting or noncommuting). We consider the problem of testing whether f is an *identity* for R, (i.e. $f(a_1, a_2, \cdots, a_m) = 0$ for all $a_i \in R$), when m is small (ideally a constant).

Given a black-box ring R by an additive generating set and a multilinear polynomial $f(x_1, x_2, \cdots, x_m)$ over R, we give a quantum algorithm of query complexity (i.e. number of oracle access) $O(k^{\frac{m}{m+1}})$, where k is the number of generators for R.

For a lower bound, we show a reduction from a variant of Collision Problem (well studied in quantum computation) to our problem. The reduction uses ideas from automata theory.

Contents

1 Introduction

The basic goal of complexity theory is to prove nontrivial upper and lower bounds for time and space required for computational problems. In proving upper and lower bounds, the representation of the input problem and the computational model play a crucial role. In algebraic complexity theory the most natural model of computation is the arithmetic circuit. An arithmetic circuit over a field is a directed acyclic graph with internal nodes labelled as addition $(+)$ and multiplication $(\times)$ gates. The circuit takes as input the indeterminates $x_1, \cdots, x_n$ and constants from a field $\mathbb{F}$. Following the addition and multiplication operations, the circuit computes a polynomial in the underlying polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$. It is not difficult to see that a small size (i.e of polynomial size) arithmetic circuit can compute a polynomial having exponential number of monomials. Also if the circuit size is s, the degree of the output polynomial can be exponential in s.

One of the central problems in algebraic complexity theory is the Polynomial Identity Testing Problem (PIT):

Given an arithmetic circuit C over a field $\mathbb{F}$ with set of indeterminates $x_1, x_2, \cdots, x_n$, can we check efficiently whether C computes the identically zero polynomial in the polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$?

A randomized polynomial-time algorithm (more precisely a co-RP algorithm) for this problem is known due to the well-known Schwartz-Zippel Lemma (see [MR01, pp.165] and [Sch80, Zip79]). The main challenge is to come up with a *deterministic* polynomial-time (or at least subexponential-time) algorithm. Over the years PIT has played a significant role in our understanding of important complexity theoretic and algorithmic problems. Well-known examples are the randomized NC algorithms for the matching problem in graphs [Lov79, MVV87], and the AKS primality test [AKS04]. The PIT problem has also played an indirect role in important complexity results such as IP $=$ PSPACE [LFK92, Sha92] and the proof of PCP theorem [ALM$^+$98]. [1] The AKS primality testing algorithm was achieved by derandomizing a *special instance* of identity testing. Due to its important connection with many central problems in complexity theory, PIT has received good attention from theoretical computer science community over the years.

The initial line of research in this area was towards reducing the number of random bits in the general PIT problem. We briefly mention the major results. The randomized identity testing algorithm based on Schwartz-Zippel Lemma is simple to describe [MR01, pp.165]: let C be the input arithmetic circuit of size s computing a polynomial $f(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ of total degree at most d. The idea is to fix a subset $S \subseteq \mathbb{F}$ of size at least $2d$ and randomly substitute $a_i \in S$ for x_i. Then if f is not an identically zero polynomial, Schwartz-Zippel Lemma guarantees that $f(a_1, a_2, \cdots, a_n) = 0$ with probability at most $1/2$. So we get a randomized polynomial-time (one sided-error) test with success probability at least $1/2$. The probability of success can be boosted up by independent repeated trials of the experiment. If we want a success probability at least $1 - \epsilon$ for a given $\epsilon > 0$, the running time and the number of random bits used in this algorithm are $(s + \log \frac{1}{\epsilon})^{O(1)}$ and $\Omega(n(\log d + \log \frac{1}{\epsilon}))$. Chen and Kao in [CK00] came up with a different idea for identity testing over the field of rational numbers $\mathbb{Q}$. There, idea is to randomly substitutes each x_i by an *irrational number* from a set of suitable size. As one can not substitute and compute a circuit over irrational numbers, they used a rational approximation technique to suitably approximate the irrationals used. The number of random bits used by their algorithm is $\sum_{i=1}^{n} \lceil \log d_i \rceil$ (d_i is the degree of x_i in f) which is better than the randomness used in Schwartz-Zippel algorithm. The running the of their algorithm is poly$(n, d, \frac{1}{\epsilon})$, where d is the total degree of the polynomial. If the size of the input circuit is s, then the total degree d can be of exponential in s, and hence the running time of their algorithm is $2^{O(s + \log \frac{1}{\epsilon})}$. Later, Lewin and Vadhan extended the algorithm of Chen-Kao to finite fields with similar running time and usage of randomness [LV98]. Finally, Agrawal and Biswas [AB03] used ideas from *Chinese Remaindering* and came up with an improved algorithm. In some sense, algorithm of Agrawal-Biswas makes a bridge between

[1] In the sense that properties of low-degree multivariate polynomials are crucial to these proofs.

the algorithms of Schwartz-Zippel and Chen-Kao-Lewin-Vadhan combining the best of them in terms of running time and randomness. The running time of their algorithm is $(s + \log \frac{1}{\epsilon})^{O(1)}$ over $\mathbb{Q}$ and $(s + \log q + \log \frac{1}{\epsilon})^{O(1)}$ over a finite field $\mathbb{F}_q$. The amount of random bits used is $\sum_{i=1}^{n} \lceil \log d_i \rceil$.

Till today, Agrawal-Biswas algorithm remains the most efficient algorithm for general PIT in terms of randomness and running time. But the algorithm is far from being deterministic polynomial-time or even deterministic subexponential-time.

Another important research direction in algebraic complexity is to prove explicit lower bounds, i.e to prove that an explicit function is hard to compute by a small size arithmetic circuit. A concrete goal is to show that the well known permanent polynomial of a $n \times n$ matrix can not be computed by a polynomial (in n) size arithmetic circuit. But just like in other computational models, the progress in proving lower bounds in arithmetic circuit model is also very limited. Nontrivial lower bounds are only known for some restricted classes of arithmetic circuits. In the case of general arithmetic circuits, the best known size lower bound is $\Omega(n \log d)$ for computing an explicit polynomial in n indeterminates and of degree d [Str73a, BS83, BCS97]. Grigoriev and Karpinski [GK98] show exponential size lower bound for depth-3 arithmetic circuits over *finite fields*. Later, the lower bound result was extended to algebras of functions over finite fields [GR98]. Shpilka and Wigderson show quadratic lower bound for depth-3 circuits over characteristic zero [SW01].

The problem of identity testing and proving lower bounds remained two seemingly different important problems in algebraic complexity theory until Impagliazzo and Kabanets showed an interesting connection between them [KI03]. Impagliazzo-Kabanets [KI03] show that giving a deterministic polynomial-time (even subexponential-time) identity testing algorithm will yield strong lower bounds. They also show a partial converse: assuming lower bounds they show a subexponential-time identity testing algorithm.

As it is widely believed that proving strong lower bounds in arithmetic circuit model is a very hard problem, the result of Impagliazzo and Kabanets suggests that derandomizing the polynomial identity testing (in general) is also a very hard problem. So the natural research direction that one can think of is to find efficient deterministic algorithms for identity testing of restricted classes of arithmetic circuits. Also, towards proving lower bounds, the first task is to prove nontrivial lower bounds for restricted classes of arithmetic circuits. In last few years, plenty of results have appeared in both directions. Dvir and Shpilka [DS06], showed a connection between construction of Locally Decodable Codes (LDC) and identity testing of depth-3 $\Sigma\Pi\Sigma$ circuits. Using explicit bounds from LDC, they were able to bound the rank of the linear forms of an identically zero $\Sigma\Pi\Sigma$ circuit. Using that, they give a polynomial-time identity testing algorithm for multilinear $\Sigma\Pi\Sigma$ circuits with bounded fan-in top Σ gate. They also show a quasi-polynomial time identity testing algorithm for $\Sigma\Pi\Sigma$ circuits with bounded fan-in top gate. Later, Kayal and Saxena improved the result giving a deterministic polynomial-time identity testing algorithm for $\Sigma\Pi\Sigma$ circuits with bounded fan-in top gate [KS07]. Their algorithm is based on application of Chinese Remaindering in *local rings*. Saxena [Sax08] show efficient identity testing algorithm for depth-3 diagonal circuits.

The algorithms of Dvir-Shpilka [DS06], Kayal-Saxena [KS07] and Saxena [Sax08] are non black-box, i.e their algorithms look into the structure of the given circuit. For the black-box case, Karnin and Shpilka [KS08] recently improve the situation giving a quasi-polynomial time identity testing algorithm for $\Sigma\Pi\Sigma$ circuits with bounded top gate fan-in. Their algorithm runs in polynomial time if the circuit is multilinear. The algorithm crucially uses the rank upper bound of identically zero $\Sigma\Pi\Sigma$ circuits obtained in [DS06] and a suitable application of an explicit extractor construction for affine sources [GR05]. Very recently, Shpilka and Volkovich consider identity testing of read-once polynomials [SV08]. An arithmetic circuit $C(x_1, x_2, \cdots, x_n)$ is a read once formula if each variable x_i occurs at most once in the leaf and C is a formula. The polynomial that C computes is called a read-once polynomial. Shpilka-Volkovich consider arithmetic circuits which are sums of k read-once formulas for a constant k. They give an algorithm for identity testing of such circuits with running time $n^{O(k^2)}$. They also consider the problem in a black-box setting. Using depth reduction technique, they give an identity testing algorithm of running time $n^{O(d+k^2)}$ where d is the depth of the circuit. In particular if d is a constant, their algorithm runs in polynomial time.

Towards proving lower bounds, in a seminal work, Raz has shown that any *multilinear formula* that computes the determinant or permanent of a $n \times n$ matrix, must be of size $n^{\Omega(\log n)}$ [Raz04]. Subsequently, using ideas similar to [Raz04], more results for explicit lower bounds are discovered [Raz06, RSY07, RY08].

More recently, a result of Agrawal and Vinay [AV08] improves our understanding about identity testing and lower bounds for general arithmetic circuits. Intuitively, their results indicates that proving strong lower bounds or (black-box) derandomization of polynomial identity testing for depth-4 arithmetic circuits is as hard as for arbitrary depth arithmetic circuits. More formally they show, if a polynomial $f(x_1, x_2, \cdots, x_n)$ of degree $d = o(n)$ can be computed using an arithmetic circuit of size $2^{o(d+d\log\frac{n}{d})}$, then f can be computed by a depth-4 circuit of similar size and the fan-in for the multiplication gates are $o(d)$. Towards identity testing they show that a complete black-box derandomization of polynomial identity testing for depth-4 circuits will give a deterministic $n^{O(\log n)}$ time identity testing algorithm for arbitrary arithmetic circuits. The proof idea is based on a depth reduction technique for arithmetic circuits introduced in [AJMV98].

All the results that we have mentioned so far are over commuting set of variables, i.e. $x_i x_j = x_j x_i$. In the noncommutative model of computation, we consider polynomials over the noncommutative ring $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ with $x_i x_j \neq x_j x_i$ (for $i \neq j$). The problem of Polynomial Identity Testing and proving lower bounds are analogously defined over noncommutative model and well studied too. The first significant size lower bound result was shown by Nisan in [Nis91]. Nisan proved that any noncommutative *arithmetic formula* for permanent must be of exponential size. So far this is the best known lower bound result in the noncommutative model.

Turning to identity testing, Bogdanov and Wee in [BW05] show a randomized polynomial time identity testing algorithm for noncommutative circuit computing small degree polynomial. Their algorithm shows a reduction from noncommutative to commutative identity testing using a well-known theorem of Amitsur and Levitzki [AL50]. Amitsur-Levitzki's theorem allows them to evaluate the given circuit over random points from a suitable small dimension matrix algebra over $\mathbb{F}$. If the output polynomial is not identically zero then the result of [AL50] guarantees that with high probability the circuit evaluates to nonzero.

Raz and Shpilka [RS05] came up with first deterministic polynomial-time identity testing algorithm for arithmetic formulas over $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$. They use ideas from [Nis91] to convert a noncommutative formula to an equivalent algebraic branching program (ABP) and give a deterministic identity test for such ABPs.

1.1 Thesis Outline

The basic goal of this thesis is to continue the study of Polynomial Identity Testing problem (and also some closely related problems) over commutative and noncommutative models and its connection with other important problems in complexity theory. Here, we highlight the main results of the thesis. Subsequently, in section 1.2, we describe the results and the organization of the thesis in detail.

A basic algebraic structure that keeps recurring in many applications related to identity testing is a finite ring (for example see [AS05] and [KS07]). It motivates the following natural question: Given an arithmetic circuit C over a finite commutative *ring* with unity, can we efficiently test whether C computes an identically zero polynomial? We give a randomized polynomial-time algorithm for this problem.

We study the connection of identity testing problem with ideal membership testing, which is another important problem in algebraic complexity theory. We show that under certain restrictions about the underlying circuits and ideals, the complexity of the two problems are closely related. A main result is a new (and possibly simpler) proof of Kayal-Saxena's identity testing algorithm for depth-3 circuits. Our proof technique departs from local ring theoretic technique (used in [KS07]) and uses simple ideas from Gröbner basis theory.

Then we turn to noncommutative polynomial identity testing problem. We show a deterministic polynomial-time identity testing algorithm for noncommutative circuits computing *sparse* and small degree polynomials. Our algorithm uses new ideas from automata theory. In fact, we give a deterministic polynomial-time algorithm to interpolate polynomials computed by such circuits.

We study the connections between the well known Isolation Lemma (introduced in [MVV87]) and circuit lower bounds. We prove that a (*non black-box*) derandomization of certain restricted version of the isolation lemma yields lower bound consequence in the noncommutative model which is analogous to the result of Impagliazzo-Kabanets [KI03]. We also show that a stronger (*black-box*) derandomization implies exponential circuit size lower bound for an explicit multilinear polynomial in commutative model.

Finally, we study the complexity of testing whether a given polynomial is an *identity* for a finite ring R (which can be commutative or noncommutative). We assume that, the input ring is given by an additive set of generators and the ring operations are performed through a ring oracle. We are interested in upper and lower bounds for the query complexity (number of accesses to the ring oracle) of the problem. In the case when the input polynomial depends only on a few variables (ideally a constant), we show a quantum algorithm of query complexity sublinear in the size of the generating set. Algorithmic ideas of our algorithm are based on the quantum algorithm for Group

Commutativity Testing by Magniez and Nayak [MN07]. For proving a lower bound, we show a reduction from a variant of the collision problem (which is well-studied in quantum computation) to our problem. To the best of our knowledge, the quantum algorithm for testing polynomial identities was never studied before.

1.2 Results and the Organization of the Thesis

Now, we describe the main results of each of the chapters of the thesis in detail. The technical contents of the thesis are organized in the following five chapters.

1.2.1 Algorithmic Problems over Finite Rings

We study some algorithmic problems over black-box rings. Black-box rings are finite rings given by a set of generators and the ring elements are encoded as binary strings. The ring operations are performed through an oracle. The black-box model for representing the finite ring is motivated by the black-box model for finite groups which is well studied in algorithmic group theory [Bab91, BS84].

The first result that we show is a randomized polynomial-time almost uniform sampling algorithm from finite rings. A crucial part of the algorithm is to compute an additive generating set for the given ring $(R, +, *)$. The input generating set for R generates R using both $*$ and $+$. The second result is a quantum polynomial-time algorithm to compute a basis representation for the finite black-box ring. [2] The algorithm uses the sampling algorithm as a subroutine to compute an additive generating set. Then, suitably using ideas from the hidden subgroup problem framework (which is well studied in quantum computation), we can compute a basis representation for R. As an application of this result, we show how to test whether a given black-box ring is a field. We give a quantum polynomial-time algorithm for this problem.

Next, we study the Polynomial Identity Testing problem over finite commutative rings with unity. We prove an analogue of Schwartz-Zippel Lemma over finite commutative rings. Suppose the input polynomial is given as a black-box. Then, using a Schwartz-Zippel analogue, we show a randomized polynomial-time algorithm for iden-

[2]For a commutative ring R, $e_1, e_2, \cdots, e_k$ is a basis if $R = \mathbb{Z}_{n_1}e_1 \oplus \mathbb{Z}_{n_2}e_2 \oplus \cdots \oplus \mathbb{Z}_{n_k}e_k$ and $e_ie_j = \sum_{\ell=1}^{k} a_{ij\ell}e_\ell$, where $a_{ij\ell} \in \mathbb{Z}_{n_\ell}$.

tity testing over finite rings satisfying some particular property. [3] More interestingly, over any *arbitrary* commutative ring with unity, we show a randomized polynomial-time identity testing algorithm when the input polynomial is given as an *arithmetic circuit*. For the underlying ring, we assume that the ring operations are performed through an oracle. The results of this chapter are reported in [ADM06, AMS08].

1.2.2 Ideal Membership and Polynomial Identity Testing

We study connections between Ideal Membership problem and Polynomial Identity Testing. The Ideal Membership Problem is the following:

Let $I = \langle g_1, g_2, \cdots, g_\ell \rangle$ be an ideal in the polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$ generated by the polynomials $g_1, g_2, \cdots, g_\ell$, where $g_i \in \mathbb{F}[x_1, x_2, \cdots, x_n]$. Also, let $f \in \mathbb{F}[x_2, x_2, \cdots, x_n]$ be a given polynomial. The ideal membership problem is to test whether $f \in I$, i.e whether there exists polynomials $a_1, a_2, \cdots, a_\ell$ in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ such that $f = a_1 g_1 + a_2 g_2 + \cdots + a_\ell g_\ell$?

We consider the input polynomial f be given by an arithmetic circuit and a monomial ideal given by a set of constant number of monomials as generators. In this case, we show that the complexity of monomial ideal membership is similar to that of polynomial identity testing. We give a randomized polynomial-time monomial ideal membership testing algorithm which is analogous to the randomized polynomial identity testing algorithm using Schwartz-Zippel Lemma.

Extending the result further, we show that if the number of generators of the given monomial ideal is not a constant, then the problem is coNP-hard. We show an upper-bound for this problem in the counting hierarchy.

We consider monomial ideal membership problem, where the input polynomial f is computed by a depth-3 $\Sigma\Pi\Sigma$ circuit with bounded fan-in output gate, and the ideal is generated by constant number of monomials. We show that the complexity of ideal membership problem in this case is essentially same as that of identity testing of $\Sigma\Pi\Sigma$ circuits with bounded fan-in output gate, which is known to be in deterministic polynomial time [KS07]. In fact the algorithm of [KS07] can be directly applied to our case. But what is more interesting is that, we can develop the algorithm and its correctness proof based on Gröbner basis theory. We believe this approach is somewhat simpler and

[3] Our algorithm works when the characteristics of the local ring components of the given ring, are suitably large depending on the degree of the given polynomial.

direct compared to the analysis of [KS07]. Moreover, this gives us a different understanding of the identity testing algorithm for such circuits. The results of this chapter are reported in [AM07].

1.2.3 Noncommutative Polynomial Identity Testing

We give a deterministic polynomial-time algorithm for testing if a sparse noncommutative polynomial having small degree (i.e the degree is bounded by a polynomial), is identically zero. The polynomial is given by an arithmetic circuit or by a black-box. Prior to our work, in noncommutative model, deterministic polynomial-time algorithm was known only for noncommutative *formulas* [RS05].

We also show a deterministic polynomial-time interpolation algorithm for sparse and small (bounded by a polynomial) degree noncommutative polynomials given by a black-box. For identity testing and interpolation, we assume that we can evaluate the given black-box polynomial over any matrix algebra.

Finally, we give an efficient deterministic reconstruction algorithm for black-box noncommutative algebraic branching programs. Our black-box model assumes that we can query for the output of *any gate* of the algebraic branching program, not just the output gate. Our algorithm is based on the algorithm of Raz and Shpilka [RS05].

The main idea behind most of the results in this part of the thesis is the selection of a small set of matrices (of small dimension) and then to evaluate the given polynomial over those matrices as input. For identity testing and interpolation, the selection of a small sized matrix family crucially uses the fact that the black-box polynomial is sparse and of small degree. The construction of such matrices uses ideas from automata theory. The results of this chapter are reported in [AMS08].

1.2.4 Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size

We study the connections between derandomizing the Isolation Lemma and proving circuit lower bounds. Originally the Isolation Lemma was formulated and proved in the seminal paper of Mulmuley, Vazirani and Vazirani [MVV87]. We briefly recall their result.

Let U be a set of size n and $\mathcal{F}$ be any collection of subsets of U. Let w be any random weight assignment to the elements of U from $[2n]$. Then the Isolation Lemma states that with high probability ($\geq 1/2$), there will be a unique minimum weight set in $\mathcal{F}$.

Over the years, this result has found several important applications in complexity theory. To state a few, a randomized NC algorithm for finding perfect matching in graphs, proving NL $\subseteq$ UL$/$poly e.t.c. Also it is known that suitably derandomizing (i.e choosing the weight function w deterministically) some restricted version of the isolation lemma will suffice to prove a NC algorithm for matching (which is an outstanding open problem in complexity theory) and NL $\subseteq$ UL.

Here we note that, in general the isolation lemma can not be derandomized completely. The reason is that there are too many (2^{2^n}) possible different set systems for U. More precisely, the following is observed in [Agr07]: The Isolation Lemma can not be fully derandomized if we allow weight functions $w : U \rightarrow [n^c]$ for a constant c (i.e. weight functions with a polynomial range). More formally, for any polynomially bounded collection of weight assignments $\{w_i\}_{i\in[n^{c_1}]}$ with weight range $[n^c]$, there exists a family $\mathcal{F}$ of $[n]$ such that for all $j \in [n^{c_1}]$, there exists two minimal weight subsets with respect to w_j. So the real question is whether we can derandomize a particular application of the isolation lemma, which is highly problem specific.

We show that derandomizing reasonably restricted versions of the isolation lemma implies circuit size lower bounds. We derive the circuit lower bounds by examining the connection between the isolation lemma and polynomial identity testing. We give a *new* randomized polynomial-time identity test for noncommutative circuits computing small degree polynomials. Our algorithm is based on the isolation lemma and automata theoretic ideas used for noncommutative sparse polynomial identity testing. Conceptually, our algorithm is very different from the algorithm of Bogdanov and Wee [BW05]. Using this result and the result of Impagliazzo-Kabanets [KI03], we show that derandomizing the isolation lemma implies noncommutative circuit size lower bounds. For the commutative case, a stronger derandomization hypothesis allows us to construct an explicit multilinear polynomial that does not have subexponential size commutative circuits. The proof technique is based on Agrawal's work on pseudorandom generator for arithmetic circuits [Agr05]. The restricted versions of the isolation lemma we consider are natural and would suffice for the standard applications of the isolation lemma. The results of this chapter are reported in [AM08].

1.2.5 Quantum Query Complexity of Multilinear Identity Testing

We study the complexity of testing whether a given multilinear polynomial is an *identity* for a given ring. To make the problem more specific, let $f(x_1, x_2, \cdots, x_m)$ be a multilinear polynomial over *any finite ring* R. The variables can be either commuting or noncommuting. Can we test efficiently whether $f(a_1, a_2, \cdots, a_m) = 0$ for all $a_i \in R$? Notice that testing whether $f(a_1, a_2, \cdots, a_m) = 0$ for all $a_i \in R$ is different from testing whether $f(x_1, x_2, \cdots, x_m)$ is an *identically zero expression* (which is the Polynomial Identity Testing Problem) for the ring $R[x_1, x_2, \cdots, x_m]$ (when x_i, x_j commute) or $R\{x_1, x_2, \cdots, x_m\}$ (when x_i, x_j do not commute). To see the difference, consider the following simple example: let R be any finite commutative ring and $R\{x_1, x_2\}$ is noncommutative polynomial ring (x_1, x_2 do not commute). Then the polynomial $f(x_1, x_2) = x_1 x_2 - x_2 x_1$ is an identity for R but it is not an identically zero expression for $R\{x_1, x_2\}$.

In general the problem of testing whether a polynomial is an identity for a ring is NP hard. In our setting, we consider the input ring R is given by a set of k additive generators and the ring operations are performed by a *ring oracle*. We are interested only in the *query complexity* of the problem i.e the number of accesses to the ring oracle.

For small m (ideally constant), we give a quantum algorithm of query complexity $O(k^{\frac{m}{m+1}})$ for testing whether $f(x_1, x_2, \cdots, x_m)$ is an identity for R. Thus the query complexity of our algorithm is sublinear in the number of generators. The algorithmic ideas of our algorithm is inspired by the quantum algorithm for group commutativity testing [MN07]. The main technical tools used in our algorithm are a generalization of a group theoretic result by Pak [Pak00] and Szegedy's results on quantum random walks [Sze04].

Towards a quantum lower bound for this problem, we show a reduction from a variant of Collision Problem (well studied in quantum computation, see [AS04]) to our problem. This reduction uses automata theoretic ideas that we have used for noncommutative polynomial identity testing. The results of this chapter are reported in [AM09].

2 Algorithmic Problems over Finite Rings

2.1 Introduction

Finite rings often play an important role in the design of algorithms for algebraic problems. Berlekamp's randomized algorithm for factoring univariate polynomials over finite fields is a classic example [Ber67, vzGG03]. More recently, as explained in [AS05], the celebrated AKS primality test [AKS04] can be cast in a ring-theoretic framework. Lenstra's survey [Len92] gives other algorithmic examples. It is shown in [AS05, KS06] that the Graph Isomorphism problem is polynomial-time many-one reducible to the Ring Isomorphism Problem. Further, it is shown in [KS06] that the Integer Factoring problem is randomized polynomial-time Turing reducible to finding an isomorphism between two finite rings.

As pointed out in [AS05], the representation of the finite ring is crucial to study algorithmic problems over finite rings. For the *polynomial representation* of finite rings it is shown that the ring automorphism problem (to test whether a given finite ring has a nontrivial automorphism) is NP-hard and the ring isomorphism problem is coNP-hard [AS05, Theorem 1]. In the *basis representation* (defined in section 2.2), the ring isomorphism problem is in NP $\cap$ coAM [KS06].

In this chapter, we explore the complexity of ring-theoretic problems where the finite rings are given by a *black-box* i.e, by a ring oracle. We give the formal definitions in Section 3.2. In a sense, a black-box ring is representation free. This model is motivated by finite black-box groups introduced by Babai and Szemerédi [BS84, Bab92] and intensively studied in algorithmic group theory.

We show a polynomial-time *quantum* algorithm to obtain a basis representation for a given black-box ring. Thus, up to quantum polynomial-time, the two representations are equivalent. A key procedure we use is an almost-uniform random sampling algorithm for finite black-box rings. Our algorithm is quite simple as compared to Babai's sampling algorithm for black-box groups [Bab91].

It is an open question whether there is a randomized polynomial-time algorithm to recover a basis representation from the black-box oracle. The main obstacle is *additive independence testing* in a black-box ring R: given $r_1, r_2, \cdots, r_k \in R$ is there a nontrivial solution to $\sum_{i=0}^{l} x_i r_i = 0$ where x_i's are integers. There is no known classical polynomial-time algorithm for this problem. However, it fits nicely into the hidden subgroup problem framework and we can solve it in quantum polynomial-time as the additive structure of R is an abelian group. As a further application of the sampling algorithm, we obtain a quantum polynomial-time algorithm for testing whether a given black-box ring is a field (Sections 2.5). The problem is inspired by primality testing: a positive integer n is prime if and only if the ring $\mathbb{Z}_n$ is a field. In primality testing the ring $\mathbb{Z}_n$ is given explicitly (because n is the input), whereas in our problem we consider black-box rings.

The second problem that we study in this chapter is the complexity of polynomial identity testing problem (PIT) (which is also the main theme of this thesis) over finite rings . We first explain the motivation behind studying the complexity of polynomial identity testing over rings. An important problem in computational algebra is ideal membership testing [Sud98, CLO92]. Suppose $I \subset \mathbb{F}[x_1, \cdots, x_n]$ is an ideal generated by polynomials $g_1, \cdots, g_r \in \mathbb{F}[x_1, \cdots, x_n]$ [1] and $f \in \mathbb{F}[x_1, \cdots, x_n]$. The ideal membership problem is to test whether $f \in I$. In general the ideal membership problem is EXPSPACE-complete [MM82, May89]. We observe a connection between ideal membership and identity testing problem. Let $I \subset \mathbb{F}[x_1, \cdots, x_n]$ be an ideal generated by polynomials $g_1, \cdots, g_r \in \mathbb{F}[x_1, \cdots, x_k]$ and $f \in \mathbb{F}[x_1, \cdots, x_n]$. Observe that $f \in I$ if and only if f is identically zero in the ring $(\mathbb{F}[x_1, \cdots, x_k]/I)[x_{k+1}, \cdots, x_n]$. Thus, ideal membership is easily reducible to polynomial identity testing when the coefficient ring is $\mathbb{F}[x_1, \cdots, x_k]/I$. Consequently, identity testing for the coefficient ring $\mathbb{F}[x_1, \cdots, x_k]/I$ is EXPSPACE-hard even when the polynomial f is given explicitly as a linear combination of monomials (we continue to study the connections between ideal membership and polynomial identity testing problem in Chapter 3).

[1]I.e $I = \{\sum_{i=1}^{r} f_i g_i \mid \forall i, f_i \in \mathbb{F}[x_1, x_2, \cdots, x_n]\}$.

This raises the following question about the complexity of PIT for the polynomial ring $R[x_1, \cdots, x_n]$, where R is a commutative ring with unity. How does the complexity depend on the structure of the ring R? We give a precise answer to this question in this chapter. We show that the algebraic structure of R is not important. It suffices that the elements of R have polynomial-size encoding, and w.r.t. this encoding the ring operations can be efficiently performed. This is in contrast to the ring $\mathbb{F}[x_1, \cdots, x_k]/I$: there are a double exponential number of elements of polynomial degree in $\mathbb{F}[x_1, \cdots, x_k]$ and the ring operations in $\mathbb{F}[x_1, \cdots, x_k]/I$ are essentially ideal membership questions and hence computationally hard.

More precisely, we study polynomial identity testing for finite commutative rings R, where we assume that the elements of R are uniformly encoded as strings in $\{0, 1\}^m$ with two special strings encoding 0 and 1, and the ring operations are carried out by queries to the *ring oracle*. In other words, we assume the ring is presented as a black-box ring. Before our work, randomized polynomial-time algorithm for identity testing was known over the ring $\mathbb{Z}_m$ for m composite [AB03].

For identity testing algorithms, we consider only finite commutative ring with unity. Although in the identity testing algorithm we do not require to use the full power of the sampling algorithm mentioned above, but a crucial part is to sample almost uniformly from the set of multiples of unity. We discuss it in more detail in the subsequent sections.

2.2 Preliminaries

A *finite ring* is a triple $(R, +, *)$, where R is a finite nonempty set such that $(R, +)$ is a commutative group and $(R, *)$ is a semigroup, such that $*$ distributes over addition. A *subring* R' is a subset of R that is a ring under the same operations. Let $S \subset R$. The subring *generated* by S is the smallest subring $\langle S \rangle$ of R containing S. Thus, if $R = \langle S \rangle$ then every element of R can be computed by an arithmetic circuit that takes as input the generators from S and has the ring operations $+$ and $*$ as the gate operations. It is easy to see that every finite ring R has a generator set of size at most $\log |R|$.

A *ring oracle* R takes queries of the form $(x, y, +)$, $(x, y, *)$, (x, addinv), where x, y are strings of *equal length* (say m) over $\Sigma = \{0, 1\}$. The response to each of these queries is either a string of length m or a symbol indicating invalid query. We assume that the additive and multiplicative identity (if any) of the ring are encoded by two special strings. Ring oracle also allows to query for the additive and multiplicative identities.

Let $R(m)$ be the set of $x \in \Sigma^m$ for which (x, addinv) is a valid query. Then R is a ring oracle if $R(m)$ is either empty or a ring with ring operations described by the responses to the above queries. The oracle R defines the finite rings $R(m)$. The subrings of $R(m)$, given by generator sets will be called *black-box rings*. To keep the description simple, without mentioning the term *black-box rings* explicitly, sometime we say that the input ring is given by a set of generators and the ring operations are performed through an oracle. Throughout this thesis, we use the terms *black-box ring* and *ring given by an oracle* interchangeably.

A *basis representation* of a finite ring R [Len92, KS06] is defined as follows: the additive group $(R, +)$ is described by a direct sum $(R, +) = \mathbb{Z}_{\ell_1}e_1 \oplus \mathbb{Z}_{\ell_2}e_2 \oplus \cdots \oplus \mathbb{Z}_{\ell_n}e_n$, where ℓ_i are the additive orders of e_i. Multiplication in R is specified by the products $e_ie_j = \sum_{k=1}^{n} \gamma_{ij}^{k} e_k$, for $1 \leq i, j \leq n$, where $\gamma_{ijk} \in \mathbb{Z}_{\ell_k}$ are the structural constants defining R.

2.2.1 Structure of Finite Rings

We recall some facts about the structure of finite commutative rings. Details can be found in excellent texts, such as [McD74, AM69]. Let R be any commutative ring. A nonempty subset I of R is an *ideal* of R if I is closed under ring addition and for all $r \in R, a \in I, ra \in I$.

A commutative ring R with unity is a *local ring* if R has a *unique* maximal ideal M. An element $r \in R$ is *nilpotent* if $r^n = 0$ for some positive integer n. An element $r \in R$ is a *unit* if it is invertible ($rr' = 1$ for some element $r' \in R$). Any element of a finite local ring is either a nilpotent or a unit. An ideal I is a *prime ideal* of R if $ab \in I$ implies either $a \in I$ or $b \in I$. For finite commutative rings, prime ideals and maximal ideals coincide. These facts considerably simplify the study of finite commutative rings (in contrast to infinite rings).

The *radical* of a finite ring R denoted by $\text{Rad}(R)$ is defined as the set of all nilpotent elements, i.e

$$\text{Rad}(R) = \{r \in R \mid \exists n > 0 \text{ s.t } r^n = 0\}$$

The radical $\text{Rad}(R)$ is an ideal of R, and it is the unique maximum ideal if R is a local ring. Let m denote the least positive integer such that for every nilpotent $r \in R$,

$r^m = 0$, i.e $(\text{Rad}(R))^m = 0$. Let R be any finite commutative ring with unity and $\{P_1, P_2, \cdots, P_\ell\}$ by the set of all maximal ideals of R. Let R_i denotes the quotient ring R/P_i^m for $1 \leq i \leq \ell$. Then, it is easy to see that each R_i is a local ring and P_i/P_i^m is the unique maximal ideal in R_i. We recall the following structure theorem for finite commutative rings.

Theorem 2.2.1 ([McD74], Theorem VI.2, page 95) *Let R be a finite commutative ring. Then R decomposes (up to order of summands) uniquely as a direct sum of local rings. More precisely*

$$R \cong R_1 \oplus R_2 \oplus \cdots \oplus R_\ell,$$

via the map $\phi(r) = (r+P_1^m, r+P_2^m, \cdots, r+P_\ell^m)$, where $R_i = R/P_i^m$ and $P_i, 1 \leq i \leq \ell$ are all the maximal ideals of R.

It is easy to see that ϕ is a homomorphism with trivial kernel. The isomorphism ϕ naturally extends to the polynomial ring $R[x_1, x_2, \cdots, x_n]$, and gives the isomorphism $\hat{\phi} : R[x_1, x_2, \cdots, x_n] \rightarrow \oplus_{i=1}^{\ell} R_i[x_1, x_2, \cdots, x_n]$.

2.2.2 Results from Quantum Computation

In this section we briefly recall some well known algorithmic results from quantum computation. These results will be useful in the proof of Theorem 2.4.1 in Section 2.4. An excellent source for details of most of these results is the text by Nielsen and Chuang [NC00].

We start by recalling the *Hidden Subgroup Problem* [NC00, pp240] (HSP): Let $G = \langle g_1, g_2, \cdots, g_k \rangle$ be a finite abelian group group and $K < G$ be a subgroup of G (which is hidden). Let X be a finite set and $f : G \rightarrow X$ be a function such that f is constant on the cosets of the subgroup K and distinct across each cosets. A standard fact in quantum computation is that a quantum algorithm can access f via a suitable oracle $U_f : |g\rangle|h\rangle \rightarrow |g\rangle|h \oplus f(g)\rangle$ for $g \in G$ and $h \in X$. Here we assume suitable binary encodings for the elements in X. The problem is to find a generating set for K.

It is well known that the Hidden Subgroup Problem can be solved in quantum polynomial-time. The algorithmic ideas are similar to the Shor's polynomial time quantum algorithms for Integer Factoring and Discrete Logarithm problem [Sho97].

A well-known application of Hidden Subgroup Problem is the *Order Finding Problem* ([NC00, pp 240-242] and [CM01]). Let G be a finite abelian black-box group (w.l.g assume additive) given by a set of generators and $a \in G$ be a given element. The Order Finding problem is to find the minimum integer r such that $ra = 0$ (r is the order of a in G). The problem of finding the order fits nicely into the framework of Hidden Subgroup Problem. To see this, let f_a be a function from $\mathbb{Z} \to G$ such that $f_a(x) = xa$. Now $f_a(x) = f_a(y)$ if and only if $x - y \in r\mathbb{Z}$. The hidden subgroup is $K = r\mathbb{Z}$. Finding the generator for K (which is r) is same as finding the order of a. Hence the Order Finding Problem can also be solved in quantum polynomial-time using the solution of Hidden Subgroup Problem.

2.3 Random Sampling from a Black-Box Ring

In this section we present a simple polynomial-time sampling algorithm that samples almost uniformly from finite black-box rings. Let R be a black-box ring generated by S. More precisely, any element of R can be generated from the elements of S using ring addition and multiplication.

We will describe a randomized algorithm that takes S as input and with high probability computes an *additive generating set* T for $(R, +)$. I.e. every element of R is expressible as a sum of elements of T.

Remark 2.3.1 *In general, for a ring R, a generating set S and an additive generating set T can be different. Consider the following example. For the ring $R = \mathbb{Z}_2[x]/\langle x^3 + x + 1\rangle$, $S = \langle 1, x\rangle$ is a generating set using $+$ and $*$. However, S is not an additive generating set.*

Using a additive generator set T for R, it turns out that we can easily sample from $(R, +)$. We start by defining formally the notion of almost uniform sampling from R.

Definition 2.3.2 *Let R be a black-box ring given by a set of generator S, and $\epsilon > 0$ be a given constant. The elements of R are uniformly encoded over $\{0, 1\}^m$. An ϵ-uniform sampling algorithm for R is a randomized polynomial-time algorithm $\mathcal{A}$ (that takes as input S, ϵ) such that for any $r \in R$:*

$$\frac{1 - \epsilon/2}{|R|} \leq \text{Prob}[\mathcal{A} \textit{ outputs } r] \leq \frac{1 + \epsilon/2}{|R|},$$

where the probability is over the internal coin tosses of $\mathcal{A}$. The algorithm $\mathcal{A}$ runs in time poly$(m, |S|, \log \frac{1}{\epsilon})$.

In the following lemma, we show an almost uniform sampling algorithm from a black-box ring R given by a set of additive generators.

Lemma 2.3.3 *Let R be a finite black-box ring given by an additive generator set $T = \{r_1, r_2, \cdots, r_n\}$ and the elements of R are encoded uniformly over $\{0,1\}^m$. Then for a given $\epsilon > 0$, there is a polynomial-time ϵ-uniform sampling algorithm for R using $O(n(m + \log \frac{1}{\epsilon}))$ ring additions and $O(n(m + \log \frac{1}{\epsilon}))$ random bits.*

Proof. Let $k_1, k_2, \cdots, k_n$ be the additive orders of $\{r_1, r_2, \cdots, r_n\}$ in R. Define the onto homomorphism $\xi : \mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_n} \longrightarrow R$ as $\xi(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} x_i r_i$. Suppose we can ϵ-uniformly sample from $\mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_n}$. Let $(x_1, x_2, \cdots, x_n)$ be a sample point from $\mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_n}$. Since ξ is an onto homomorphism, $\xi^{-1}(r)$ has the same cardinality for each $r \in R$. More precisely, the cardinality of $\xi^{-1}(r)$ is equal to the cardinality of the kernel of the homomorphism ξ. Hence, $\xi(x_1, x_2, \cdots, x_n)$ is an ϵ-uniformly distributed random element from R.

Thus, it suffices to show that we can almost uniformly sample from $\mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_n}$. Notice that we do not know the k_i's. But we know an upper bound, namely 2^m, for each of $k_1, k_2, \ldots, k_n$. Take a suitably large $M > 2^m$ to be fixed later in the analysis. The sampling is as follows: pick $(x_1, x_2, \cdots, x_n)$ uniformly at random from $[M]^n$ and output $\sum x_i r_i$. Let $(a_1, a_2, \cdots, a_n) \in \mathbb{Z}_{k_1} \times \mathbb{Z}_{k_2} \times \cdots \times \mathbb{Z}_{k_n}$ and let,

$$P = \text{Prob}[x_i \equiv a_i \bmod k_i, 1 \leq i \leq n].$$

The x_i for which $x_i \equiv a_i \bmod k_i$ are precisely $a_i, a_i + k_i, \cdots, a_i + k_i \lfloor (M - a_i)/k_i \rfloor$. Let $M_i' = \lfloor (M - a_i)/k_i \rfloor$. Then $P = \prod_{i=1}^{n} \frac{M_i'+1}{M} \leq \prod_{i=1}^{n} \frac{1+\frac{M-a_i}{k_i}}{M} = \prod_{i=1}^{n} \frac{1}{k_i}(1 + \frac{k_i - a_i}{M})$. Using the fact that $k_i \leq 2^m$, we get $P \leq \prod_{i=1}^{n} \frac{1}{k_i}(1 + \frac{2^m}{M})$.

On the other hand, it is easy to check that, $P \geq \frac{1}{M^n} \prod_i (\frac{M-a_i}{k_i}) \geq \prod_i (1/k_i)(1 - \frac{a_i}{M}) \geq (1 - \frac{2^m}{M}) \prod_i \frac{1}{k_i}$. Choosing $M = \lceil (2^{m+1})/\epsilon \rceil$, we can see that,

$$\frac{1 - \frac{\epsilon}{2}}{|R|} \leq P \leq \frac{1 + \frac{\epsilon}{2}}{|R|}.$$

The number of ring additions required is $O(n \log M)$ which is of $O(n(m + \log \frac{1}{\epsilon}))$. To compute $x_i r_i$ we use a standard doubling algorithm and thus make at most $O(\log x_i)$ queries to the ring oracle, which is bounded by $O(\log M)$. Also it is easy to see that the number of random bits used is of $O(n(m + \log \frac{1}{\epsilon}))$. ∎

Let $R = \langle S \rangle$ be a black-box ring. Denote by $\hat{R}$ the additive subgroup of $(R, +)$ generated by S. I.e. $\hat{R}$ is the smallest additive subgroup of $(R, +)$ containing S. Notice that $\hat{R}$ could be a proper subset of R, and $\hat{R}$ need not be a subring of R in general.

Lemma 2.3.4 *Let $R = \langle S \rangle$ be a black-box ring, and $\hat{R}$ be the additive subgroup of $(R, +)$ generated by S. Then $\hat{R} = R$ if and only if $\hat{R}$ is closed under the ring multiplication: i.e. $\hat{R}r \subseteq \hat{R}$ for each $r \in S$.*

Proof. If $\hat{R} = R$ then the condition is obviously true. Conversely, notice that $\hat{R}r \subseteq \hat{R}$ for each $r \in S$ implies that $\hat{R}$ is closed under multiplication and hence $\hat{R} = R$. ■

Theorem 2.3.5 *There is a randomized algorithm that takes as input a black-box ring $R = \langle S \rangle$ (the elements of R are uniformly encoded over $\{0, 1\}^m$) and with high probability computes an additive generating set for $(R, +)$ and runs in time polynomial in the input size.*

Proof. The algorithm starts with S and proceeds in stages by including new randomly picked elements into the set at every stage. Thus, it computes a sequence of subsets $S = S_1 \subseteq S_2 \subseteq \ldots \subseteq S_\ell$, where ℓ will be appropriately fixed in the analysis. Let H_i denote the additive subgroup generated additively by S_i for each i. Notice that $H_1 = \hat{R}$. We now describe stage i of the procedure where, given S_i, the algorithm will compute S_{i+1}. First, notice that for each $r \in S$, $H_i r$ is a subgroup of $(R, +)$ that is additively generated by $\{xr \mid x \in S_i\}$. Thus, we can use Lemma 2.3.3 to ϵ-uniformly sample (for a given ϵ) in polynomial time an element x_{ir} from $H_i r$, for each $r \in S$. We now define the set $S_{i+1} = S_i \cup \{x_{ir} \mid r \in S\}$. Clearly, if ℓ is polynomially bounded then the above sampling procedure outputs S_ℓ in polynomial time. It thus remains to analyze the probability that S_ℓ additively generates $(R, +)$.
Claim. For $\ell = 4m + 1$ the probability that S_ℓ additively generates $(R, +)$ is at least $1/3 - \delta(\epsilon)$, where $\delta(\epsilon)$ will be appropriately fixed in the analysis.
Proof of Claim:. The proof is a simple application of Markov inequality. We first recall Markov inequality from [MR01, pp 46].

Markov Inequality: Let Y be a random variable that takes only non-negative values. Then for any positive number t,

$$\text{Prob}[Y \geq t] \leq \frac{E[Y]}{t}.$$

$E[Y]$ is the expectation of the random variable Y.

We define indicator random variables $Y_i, 1 \leq i \leq 4m$ as follows: $Y_i = 1$ if $H_i = H_{i+1}$ and $H_i \neq R$, and $Y_i = 0$ otherwise. Let $Y = \sum_{i=1}^{4m} Y_i$. First, we bound the expected value of each Y_i. If $H_i = R$ then clearly $E[Y_i] = 0$. Suppose $H_i \neq R$. By Lemma 2.3.4, there is an $r \in S$ such that $H_i r \not\subseteq H_i$. As $H_i r$ is an additive group it follows that $H_i r \cap H_i$ is a proper subgroup of $H_i r$ and hence $|H_i r \cap H_i| \leq |H_i r|/2$. Therefore, for a ϵ-uniformly sampled $x \in H_i r$ the probability that it lies in H_i is at most $1/2(1+\epsilon/2)$ (use Lemma 2.3.3). So, Prob$[Y_i = 1] \leq$ Prob$[x_{ir} \in H_i] \leq 1/2(1+\epsilon/2)$ Putting it together, we get $\mu = E[Y] \leq 2m(1+\epsilon/2)$. Now, by Markov inequality Prob$[Y \geq 3m] \leq \mu/3m \leq 2/3(1+\epsilon/2)$.

So, for the proof of the claim, we take $\delta(\epsilon) = \epsilon/3$. ■

Combining Theorem 2.3.5 with Lemma 2.3.3, we immediately obtain the main theorem of this section.

Theorem 2.3.6 *There is a polynomial-time almost uniform sampling algorithm from black-box rings that takes as input $R = \langle S \rangle$ and $\epsilon > 0$, runs in time polynomial in input size and $\log(1/\epsilon)$ and outputs an ϵ-uniform random element from the ring R.*

Let $R = \langle r_1, r_2, \ldots, r_n \rangle$ be a black-box ring with elements encoded as strings in Σ^m. Examining the proof of Theorem 2.3.5 it is easy to see that every element $r \in R$ can be computed by an arithmetic circuit C_r that takes as input the generators $r_1, r_2, \ldots, r_n$ and has gates labelled $+$ and $*$ corresponding to the ring operations, such that C_r evaluates to r, and the size of the circuit C_r is $O(m^3n^2)$. This is analogous to the reachability lemma for finite black-box groups [BS84]. We briefly explain the proof.

Lemma 2.3.7 (ring reachability lemma) *Let $R = \langle r_1, r_2, \ldots, r_n \rangle$ be a black-box ring with elements encoded as strings in Σ^m. For every $r \in R$ there is an arithmetic circuit C_r of size $O(m^3n^2)$ that has gates labelled by ring operations $+$ and $*$, takes as input $r_1, r_2, \ldots, r_n$ and evaluates to r.*

Proof. Let $S = \langle r_1, r_2, \cdots, r_n \rangle$. We analyze the proof of Theorem 2.3.5 to construct the arithmetic circuit C_r. It is convenient to think of C_r as having two part $C_r^{'}$ and $C_r^{''}$. $C_r^{'}$ takes as input m and generators $r_i \in S$ and compute an additive set of generators T for R. The input to $C_r^{''}$ are the generators from T and the output is r. From the Theorem 2.3.5, it is clear that $|T| = O(mn)$. So $C_r^{''}$ can be easily implemented using

$O(m^2n)$ gates. To find the size of $C_r^{'}$, we carefully analyze the number of arithmetic operations required to construct set S_{i+1} from S_i. The number of generators in S_i is at most ni. We need to compute n elements from the groups H_ir where $r \in S_1$ (recall that $S_1 = S$). More precisely, we need to compute n sums of the form $\sum_{r_i \in S_i} x_i r_i$ and $x_i \leq 2^m$. This can be easily implemented using $O(mn^2i)$ arithmetic operations [2]. Thus the total number of arithmetic operations involved $\sum_{i=1}^{O(m)} O(mn^2i)$, which is $O(m^3n^2)$. This completes the proof. ■

2.4 Quantum Algorithm for Finding a Basis Representation

In this section we describe a quantum polynomial-time algorithm that takes a black-box ring and computes a basis representation for it. The algorithm is Monte Carlo with small error probability.

Theorem 2.4.1 *There is a quantum polynomial-time algorithm that takes a black-box ring as input and computes a basis representation for the ring with small error probability.*

Proof. Let $R = \langle S \rangle$ be the input black-box ring. By the algorithm in Theorem 2.3.5 we first compute an additive generating set $\{r_1, r_2, \cdots, r_n\}$ for R. Then we compute the additive orders $\{d_1, d_2, \cdots, d_n\}$ of $\{r_1, r_2, \cdots, r_n\}$ in the abelian group $(R, +)$. This step nicely fits into the hidden subgroup problem framework which is well studied in quantum computation. To be more precise, efficient quantum solution is known for the following problem [Mos99]:

Let $G = \langle g_1, g_2, \cdots, g_n \rangle$ be a finite abelian black-box group such that the elements of G are uniformly encoded over $\{0, 1\}^m$. Then, given an element $a \in G$, one can find the order of a in G in quantum polynomial-time. [3]

The algorithmic ideas for solving this problem are similar to the quantum polynomial-time algorithm for integer factoring [Sho97] (see the discussion in section 2.2.2). Mosca's PhD thesis gives a self-contained solution for this problem [Mos99, Section 2.8.2, 2.8.3].

[2]We always use standard doubling algorithm to compute $x_i r_i$.

[3]Notice that we do not know $|G|$ but we know an upper bound that $|G| \leq 2^m$.

The next step is to extract an *additively independent* set T of generators from the additive generating set $\{r_1, r_2, \cdots, r_n\}$. A generating set $\{e_1, e_2, \cdots, e_k\}$ for R is an *additively independent* generating set if $R = \mathbb{Z}_{n_1}e_1 \oplus \mathbb{Z}_{n_2}e_2 \oplus \cdots \oplus \mathbb{Z}_{n_k}e_k$ where n_i's are the additive orders of e_i's. Recall from the definition of basis representation of a finite ring (defined in section 2.2) that such a generating set always exists.

The way to compute such a generating set is well known in quantum computation. Details can be found in in [CM01, section 5] and also in Mosca's PhD thesis [Mos99, section 2.8.3]. The idea is to first decompose $(R, +)$ as the direct sum of its Sylow subgroups. This decomposition uses ideas similar to Shor's algorithm [Sho97]. Then each of the Sylow subgroups can further be decomposed into direct sum of cyclic groups by solving instances of the hidden subgroup problem.

Let the additively independent set of generators computed in quantum polynomial-time (as explained) be $T = \{\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_\ell\}$. To get the structural constants of the basis representation it remains to express the products rr', for $r, r' \in T$, as integer linear combinations of elements of T. We can again use Shor's period-finding quantum algorithm to compute the additive order d of rr'. Let $\{\hat{d}_1, \hat{d}_2, \cdots, \hat{d}_\ell\}$ are the orders of the generators in T. Then, we define a homomorphism $\varphi : \mathbb{Z}_d \times \mathbb{Z}_{\hat{d}_1} \times \cdots \times \mathbb{Z}_{\hat{d}_\ell} \to (R, +)$ as $\varphi(a, a_1, a_2, \cdots, a_\ell) = -arr' + \sum_{j=1}^{\ell} a_j \hat{r}_j$. Again using the polynomial-time quantum algorithm for Hidden Subgroup Problem [IMS03], we can find an additive generating set for Ker(φ). Let M be the integer matrix whose columns are the generators of Ker(φ) (here we think of the column vectors as integer vectors). The next step is to compute a basis for Ker(φ) from its set of generators. We use ideas similar to that described in [CM01, Theorem 7]. Let M_h be the corresponding *Hermite Normal Form* for M that can be computed in deterministic polynomial time (see [Sch98, pp 45-59] or [KB79]). The integer linear span of the column vectors of M_h is same as that of M.

Let the $(1, 1)^{\text{th}}$ entry of M_h be u and the column vectors of M_h are $v_1, v_2, \cdots, v_\ell$. Any vector v in the column space of M_h looks like, $v = \sum_{i=1}^{\ell} \mu_i v_i$, where $\mu_i \in \mathbb{Z}$ and the 1^{th} entry of v is $\mu_1 u$.

To express rr' as a integer linear combination of the basis elements, it suffices to seek for a vector of the form $(w, x_1, x_2, \cdots, x_\ell)$ in the column space of M$_h$, where $w \equiv 1 \bmod d$. Thus it is necessary that, the $(1, 1)^{\text{th}}$ entry of M$_h$ (which is u) has to be an invertible element in the ring $\mathbb{Z}_d$. Let λ be the inverse of u modulo d. I.e $u\lambda \equiv 1 \bmod d$. If C_1 is the first column of M_h, it is easy to see that λC_1 is a solution of the form $(1, y_1, y_2, \cdots, y_\ell)$ in the ring $\mathbb{Z}_d \times \mathbb{Z}_{\hat{d}_1} \times \cdots \times \mathbb{Z}_{\hat{d}_\ell}$, using which we can express

rr' as an integer linear combination of the basis elements. ■

2.5 Testing if a Black-Box Ring is a Field

In this section we describe a simple quantum polynomial-time algorithm that takes a black-box ring as input and tests if it is a field. This result can be seen as a sort of generalization of primality testing: the ring $\mathbb{Z}_n$ is a field if and only if n is a prime. However, the black-box setting for the problem presents obstacles, like finding the additive order of elements in a finite ring, that seem hard for classical (randomized) polynomial-time computation.

Theorem 2.5.1 *There is a quantum polynomial-time algorithm with small error probability for testing if a given black-box ring is a field.*

Proof. Let R be the input black-box ring. Applying the quantum polynomial-time algorithm in Theorem 2.4.1 we obtain with high probability a basis representation for R: $(R, +) = \mathbb{Z}_{m_1}e_1 \oplus \mathbb{Z}_{m_2}e_2 \oplus \cdots \oplus \mathbb{Z}_{m_n}e_n$, and $e_ie_j = \sum_{k=1}^{n} \gamma_{ijk}e_k, \gamma_{ijk} \in \mathbb{Z}_{m_k}$.

Clearly, R is a field only if all m_i's are equal to a prime p. Using the AKS primality testing [AKS04] (or one of the polynomial-time randomized tests) we check if p is prime. If not then the input is rejected. Thus, the basis representation can be written as $(R, +) = \mathbb{F}_pe_1 \oplus \mathbb{F}_pe_2 \oplus \cdots \oplus \mathbb{F}_pe_n$.

We next compute the minimal polynomial of e_1 over $\mathbb{F}_p$. [4] This can be easily done in deterministic polynomial time. We explain it briefly. Notice that $|R| = p^n$, so the degree of the minimal polynomial of e_1 is less than n.

To check whether e_1 has a minimal polynomial of degree j, we seek for a solution $a_0, a_1, \cdots, a_{j-1} \in \mathbb{F}_p$ such that $e_1^j = a_0 + a_1e_1 + \cdots + a_{j-1}e_1^{j-1}$. Now using the relations for the structural constrains, we can write the powers of e_1 as $\mathbb{F}_p$-linear combinations of $e_1, e_2, \cdots, e_n$. Also, one can easily identify the representation of the unity (1) in R as a linear combination of $e_1, e_2, \cdots, e_n$. To find that, write $1 = x_1e_1 + x_2e_2 + \cdots + x_ne_n$, where $x_i \in \mathbb{F}_p$ are the unknowns. Now for $i \in \{1, 2, \cdots, n\}$, we multiply both sides of the equation by e_i and use the relations e_ie_j to construct the following equation:

$$L_1^ie_1 + L_2^ie_2 + \cdots + L_n^ie_n = 0,$$

[4]The minimal polynomial $m(x) \in \mathbb{F}_p[x]$ of an element e over the field $\mathbb{F}_p$ is a degree d polynomial such that $m(e) = 0$ and e does not satisfy any polynomial of degree less than d.

where L_s^i are the linear forms in $x_1, x_2, \cdots, x_n$ over $\mathbb{F}_p$. Now, setting $L_s^i = 0$ for $s \in \{1, 2, \cdots, n\}$ and $i \in \{1, 2, \cdots, n\}$ gives linear constraints for $x_1, x_2, \cdots, x_n$, which we can easily solve.

Now, the problem of finding the unknowns $a_0, a_1, \cdots, a_{j-1}$ (we think $a_0 = a_0 \cdot 1$ and use the representation of 1 in terms of the basis elements) boils down to solving a system of linear equations over $\mathbb{F}_p$, which can be done in deterministic polynomial time. To compute a minimal polynomial for e_1, we need to find least $j \in [n-1]$.

Suppose the minimal polynomial for e_1 is $m_1(x)$ with degree d_1. Then in deterministic polynomial time we can test if $m_1(x)$ is irreducible over $\mathbb{F}_p$ [vzGG03]. If it is not then the input R is rejected. Otherwise,

$$\mathbb{F}_p(e_1) = \{a_0 + a_1 e_1 + a_2 e_1^2 + \cdots + a_{d_1-1} e_1^{d_1-1} \mid \text{for } 1 \leq i \leq d_1 - 1,\ a_i \in \mathbb{F}_p\}.$$

$\mathbb{F}_p(e_1)$ is isomorphic to the finite field $\mathbb{F}_p[x]/\langle m_1(x)\rangle$ which is isomorphic to $\mathbb{F}_{p^{d_1}}$. Also, notice that $\{1, e_1, \cdots, e_1^{d_1-1}\}$ is a basis for the field $\mathbb{F}(e_1)$ and each of the basis elements can be expressed as a linear combination of $e_1, e_2, \cdots, e_n$ using the pairwise multiplication relations between e_i's.

With the above step as the base case, inductively we assume that at the i-th step of the algorithm we have computed the finite field $\mathbb{F} = \mathbb{F}_p(e_1, e_2, \ldots, e_i)$ contained in R with a basis $\{v_1, v_2, \cdots, v_k\}$ for $\mathbb{F}$ over $\mathbb{F}_p$, where each v_i is expressed as a $\mathbb{F}_p$-linear combination of $\{e_1, e_2, \cdots, e_n\}$. Let $d = \prod_{t=1}^{i} d_t$, where d_t is the degree of the minimal polynomial of e_t over $\mathbb{F}(e_1, \ldots, e_{t-1})$ for each t. By induction hypothesis $\mathbb{F}_p(e_1, e_2, \cdots, e_i) \cong \mathbb{F}_{p^d}$. As $v_1, v_2, \cdots, v_k$ is a basis for $\mathbb{F}_p(e_1, \ldots, e_i)$, it is clear that $\mathbb{F}_p(e_1, \ldots, e_i) \cong \mathbb{F}_{p^k}$. So $k = d$.

Proceeding inductively, at the $i+1$-th step we again compute the minimum polynomial $m_{i+1}(x)$ of e_{i+1} over the field $\mathbb{F} = \mathbb{F}_p(e_1, e_2, \cdots, e_i)$. To check whether e_{i+1} has a minimal polynomial of degree j over $\mathbb{F}$, write,

$$e_{i+1}^j = \sum_{\ell=0}^{j-1} L_\ell(v_1, v_2, \cdots, v_d) e_{i+1}^\ell,$$

where L_ℓ's are linear forms in $v_1, v_2, \cdots, v_d$ over $\mathbb{F}_p$, i.e $L_\ell(v_1, v_2, \cdots, v_d) = \sum_{j=1}^{d} x_{\ell j} v_j$, where $x_{\ell j}$ are the unknowns. Notice that, by induction hypothesis, we already have the representations of v_s (for $1 \leq s \leq d$) as linear forms in $e_1, e_2, \cdots, e_n$ over $\mathbb{F}_p$.

Now, following the algorithm already mentioned above, it is easy to see that this computation will also boil down to solving a system of linear equations over $\mathbb{F}_p$. Also, we will similarly be able to check in polynomial time whether the obtained minimal polynomial is irreducible over $\mathbb{F}_{p^d}$ [vzGG03].

We continue this procedure for n steps and if in none of the steps the above algorithm rejects the input, we conclude that R is a field. Clearly, if the basis representation for R is correct (which it is with high probability), the algorithm will correctly decide. ■

In the above theorem the power of quantum computation is used only to recover a basis representation for R. If R is already in basis representation then field testing is in P. We now give a classical complexity upper bound for the field testing.

Theorem 2.5.2 *Testing if a black-box ring is a field is in* AM ∩ coNP.

Proof. A finite ring $R = \langle r_1, \ldots, r_n \rangle$ is not a field if and only if it has zero divisors: nonzero elements $a, b \in R$ whose product $ab = 0$. An NP test for this would be to guess small circuits C_a and C_b (using Lemma 2.3.7) for zero divisors a and b verifying their product $ab = 0$ using the black-box oracle. Thus the problem is in coNP.

We now show that the problem is in AM. Our goal is to compute a basis representation for R in AM. The rest is clear from the proof of Theorem 2.5.1.

Merlin will send a basis representation for R to Arthur as follows: Merlin sends a basis $\{u_1, u_2, \cdots, u_\ell\}$ for $(R, +)$ in terms of the given set of generators. More precisely, Merlin sends $\{u_1, u_2, \cdots, u_\ell\}$ as small arithmetic circuits which take as inputs the generators $r_1, r_2, \cdots, r_n$. Again, this is always possible due to Lemma 2.3.7. Also, Merlin sends each generator r_i as a linear combination of the elements u_i's and the pairwise product relations $u_i u_j$ as a linear combination of u_i's.

Arthur can now easily verify that $\{u_1, u_2, \cdots, u_l\}$ is a generating set for R. For that Arthur verifies that each of the generators r_i can be obtained from the linear combination of u_j's (that Merlin has sent) correctly and that the product relations $u_i u_j$ are correct. Arthur needs to make queries to the ring oracle.

It remains to verify that $\{u_1, u_2, \cdots, u_l\}$ is *additively independent*. Merlin sends the additive order p of u_i's. Using AKS primality testing [AKS04], Arthur verifies that p is prime and checks $\forall i : pu_i = 0$. Now, to verify that $\{u_1, u_2, \cdots, u_\ell\}$ are additively independent it suffices to check that the additive group $(R, +) = \langle u_1, u_2, \cdots, u_\ell \rangle$ is of order p^ℓ. By a result of Babai [Bab92], order verification of black-box groups is in AM. This protocol can clearly be applied to G. ■

2.6 Schwartz-Zippel Lemma over Finite Rings

In the rest of the sections of this chapter, we study the Polynomial Identity Testing (PIT) over finite commutative ring with unity. We start by defining Polynomial Identity Testing problem over finite ring.

Let R be a finite commutative black-box ring with unity. Let C be a given arithmetic circuit with internal nodes labelled by $+$ and $*$ gates, that takes as inputs indeterminates $x_1, x_2, \cdots, x_n$ and elements from the ring R. Using the $+$ and $*$ gates C computes a polynomial $f(x_1, x_2, \cdots, x_n)$ in $R[x_1, x_2, \cdots, x_n]$. The identity problem is, given C and an oracle for R, test whether $f \equiv 0$ in $R[x_1, x_2, \cdots, x_n]$.

In this section, we first give a generalization of Schwartz-Zippel Lemma [MR01] to finite commutative rings and apply it for identity testing of black-box polynomials in $R[x_1, \cdots, x_n]$, where R is a finite commutative ring with unity whose elements are uniformly encoded by strings from $\{0,1\}^m$ with a special string e denote unity, and the ring operations are performed by a ring oracle.

2.6.1 The Schwartz-Zippel Lemma

We observe the following easy fact about zeros of univariate polynomials over finite commutative rings with unity.

Proposition 2.6.1 *Let R be a finite commutative ring with unity 1 containing a field $\mathbb{F}$ such that $1 \in \mathbb{F}$. If $f \in R[x]$ is a nonzero polynomial of degree d then $f(a) = 0$ for at most d distinct values of $a \in \mathbb{F}$.*

Remark 2.6.2 *It is easy to find examples of finite commutative ring R with unity 1 such that R contains a field F and $1 \in \mathbb{F}$. Consider the following example. Let $R = \mathbb{F}_p \times \mathbb{F}_p$ for a prime p. Notice that R is not an integral domain as it has zero-divisors ($(1,0)\cdot(0,1) = (0,0)$). Consider the set of elements $S = \{(i,i) \mid 0 \le i \le p-1\}$. It is easy to see that S forms a field under ring addition and multiplication.*

Proof. (of Proposition 2.6.1)

Suppose $a_1, a_2, \cdots, a_{d+1} \in \mathbb{F}$ are distinct points such that $f(a_i) = 0, 1 \leq i \leq d+1$. Then we can write $f(x) = (x - a_1)q(x)$ for $q(x) \in R[x]$. Dividing $q(x)$ by $x - a_2$ yields $q(x) = (x - a_2)q'(x) + q(a_2)$, for some $q'(x) \in R[x]$. Thus, $f(x) = (x - a_1)(x - a_2)q'(x) + (x - a_1)q(a_2)$. Putting $x = a_2$ in this equation gives $(a_2 - a_1)q(a_2) = 0$. But $(a_2 - a_1)$ is nonzero in the field $\mathbb{F}$ and hence is invertible and $(a_2 - a_1)^{-1}(a_2 - a_1) = 1$. Multiplying by $(a_2 - a_1)^{-1}$ we get $q(a_2) = 0$. Consequently, $f(x) = (x - a_1)(x - a_2)q'(x)$ in $R[x]$. Applying this argument successively for the other a_i finally yields $f(x) = g(x)\prod_{i=1}^{d+1}(x - a_i)$ for some nonzero polynomial $g(x) \in R[x]$. Since $\prod_{i=1}^{d+1}(x - a_i)$ is a monic polynomial, this forces $\deg(f) \geq d + 1$ which is a contradiction. ∎

Using Proposition 2.6.1 we describe an easy generalization of the Schwartz-Zippel lemma to finite commutative rings with unity containing a field.

Lemma 2.6.3 *Let R be a finite commutative ring with 1 such that R contains a field $\mathbb{F}$ with $1 \in \mathbb{F}$. Let $g \in R[x_1, x_2, \cdots, x_n]$ be any polynomial of degree at most d. If $g \not\equiv 0$, then for any subset A of $\mathbb{F}$ we have*

$$\text{Prob}_{a_1 \in A, \cdots, a_n \in A}[g(a_1, a_2, \cdots, a_n) = 0] \leq \frac{d}{|A|}.$$

Proof. We need to show that the number of n-tuples $(a_1, \cdots, a_n) \in A^n$ such that $g(a_1, a_2, \cdots, a_n) = 0$ is at most $d|A|^{n-1}$. The proof is by induction on n. The base case $n = 1$ involves a univariate polynomial $g(x_1)$ in $R[x_1]$ and follows directly from Proposition 2.6.1. As induction hypothesis suppose the lemma holds for multivariate polynomials in $n - 1$ indeterminates. Write $g(x_1, x_2, \cdots, x_n)$ as $g(x_1, x_2, \cdots, x_n) = \sum_{i=0}^{k} x_n^i g_i(x_1, x_2, \cdots, x_{n-1})$, where $k \leq d$ is the largest exponent of x_n in g with nonzero coefficient g_k, and each $g_i \in R[x_1, x_2, \cdots, x_{n-1}]$. Since $g_k \neq 0$ and $\deg(g_k) \leq d-k$, by the induction hypothesis there are at most $(d-k)|A|^{n-2}$ tuples $(a_1, \cdots, a_{n-1}) \in A^{n-1}$ such that $g_k(a_1, \cdots, a_{n-1}) = 0$. Let

$$E_1 = \{(a_1, \cdots, a_n) \mid g_k(a_1, \cdots, a_{n-1}) = 0\}.$$

Then $|E_1| \leq (d-k)|A|^{n-1}$. Now consider the univariate polynomial,

$$\hat{g}(x_n) = \sum_{i=0}^{k} x_n^i g_i(a_1, a_2, \cdots, a_{n-1})$$

in $R[x_n]$ for $(a_1, \cdots, a_{n-1}) \in A^{n-1}$.

If $g_k(a_1, a_2, \cdots, a_{n-1})$ is nonzero then $\hat{g}(x_n)$ is a nonzero polynomial. Let

$$E_2 = \{(a_1, \cdots, a_n) \mid \hat{g}(x_n) \neq 0 \text{ and } \hat{g}(a_n)) = 0\}.$$

It follows from Proposition 2.6.1 that $|E_2| \leq k|A|^{n-1}$.

Since $\{(a_1, \cdots, a_n) \mid g(a_1, \cdots, a_n) = 0\} \subseteq E_1 \cup E_2$, we obtain the required bound

$$|\{(a_1, \cdots, a_n) \mid g(a_1, \cdots, a_n) = 0\}| \leq |E_1| + |E_2| \leq (d-k)|A|^{n-1} + k|A|^{n-1} = d|A|^{n-1}.$$

This completes the proof. ■

In general Lemma 2.6.3 is not useful, because the given finite ring R may not contain a large finite field $\mathbb{F}$ containing 1. We explain how to get around this problem for finite commutative local rings. Because of the structure theorem 2.2.1, it suffices to consider local rings.

Let R be a finite local ring with unity given by a ring oracle. Suppose the characteristic of R is p^α for a prime p. If the elements of R are encoded in $\{0,1\}^m$ then 2^m upper bounds the size of R. Let $M > 2^m$, to be fixed later in the analysis. Let $U = \{ce \mid 0 \leq c \leq M\}$, where e denotes the unity of R. We will argue that, for a suitable M, if we sample ce uniformly from U then $(c \bmod p)e$ is almost uniformly distributed in $\mathbb{Z}_p e = \{xe | 0 \leq x \leq p-1\}$. Pick x uniformly at random from $[0, 1, \cdots, M-1]$ and output xe. Let $a \in \mathbb{Z}_p$ and $P = \text{Prob}[x \equiv a \bmod p]$. The x for which $x \equiv a \bmod p$ are $a, a+p, \cdots, a+p\lfloor \frac{M-a}{p} \rfloor$. Let $M' = \lfloor \frac{M-a}{p} \rfloor$. Then $P = M' + 1/M \leq \frac{1}{p}(1 + \frac{2^m}{M})$. Clearly, $P \geq \frac{1}{p}(1 - \frac{2^m}{M})$. For a given $\epsilon > 0$, choose $M = 2^{m+1}/\epsilon$. Then $\frac{1-\epsilon/2}{p} \leq P \leq \frac{1+\epsilon/2}{p}$. So $(x \bmod p)e$ is $\frac{\epsilon}{2}$-uniformly distributed in $\mathbb{Z}_p e$.

Lemma 2.6.4 *Let R be a finite local commutative ring with unity 1 and of characteristic p^α for a prime p. The elements of R are encoded using binary strings of length m. Let $g \in R[x_1, x_2, \cdots, x_n]$ be a polynomial of degree at most d and $\epsilon > 0$ be a given constant.*

If $g \not\equiv 0$, *then*

$$\text{Prob}_{a_1 \in U, \cdots, a_n \in U}[g(a_1, a_2, \cdots, a_n) = 0] \leq \frac{d}{p}(1 + \frac{\epsilon}{2}),$$

where $U = \{ce \mid 0 \leq c \leq M\}$ *and* $M > 2^{m+1}/\epsilon$.

Proof. Consider the following tower of ideals inside R :

$$R \supseteq pR \supseteq p^2R \supseteq \cdots \supseteq p^\alpha R = \{0\}.$$

Let k be the integer such that $g \in p^k R[x_1, \cdots, x_n] \setminus p^{k+1}R[x_1, \cdots, x_n]$. Write $g = p^k \hat{g}$. Consider the ring, $\hat{I} = \{r \in R \mid p^k r = 0\}$. Clearly, $\hat{I}$ is an ideal of R. Let $S = R/(\hat{I} + pR)$ which is a finite commutative ring with unity $1 + (\hat{I} + pR)$.

We claim that $\hat{g}$ is a nonzero polynomial in $S[x_1, \cdots, x_n]$. Otherwise, let $\hat{g} \in (\hat{I} + pR)[x_1, \cdots, x_n]$. Write $\hat{g} = g_1 + g_2$, where $g_1 \in \hat{I}[x_1, \cdots, x_n]$ and $g_2 \in pR[x_1, \cdots, x_n]$. Then $p^k\hat{g} = p^k g_2$ as $p^k g_1 = 0$. But $g_2 \in pR[x_1, \cdots, x_n]$, which contradicts the fact that k is the largest integer such that $g \in p^k R[x_1, \cdots, x_n]$. Thus $\hat{g}$ is a nonzero polynomial in $S[x_1, \cdots, x_n]$. Now we argue that S contains the finite field $\mathbb{F}_p$, and then using the Lemma 2.6.3, the proof of the lemma will follow easily. To see a copy of $\mathbb{F}_p$ inside S, it is enough to observe that $\{i + (\hat{I} + pR) \mid 0 \leq i \leq p - 1\}$ as a field is isomorphic to $\mathbb{F}_p$ and contains the unity of S. Clearly the failure probability for identity testing of g in $R[x_1, \cdots, x_n]$ is upper bounded by the failure probability for the identity testing of $\hat{g}$ in $S[x_1, \cdots, x_n]$. Consider the natural homomorphism $\phi : U \to \mathbb{F}_p$, given by $\phi(ce) = c \bmod p$. Thus if we sample uniformly from U, using ϕ, we can $\frac{\epsilon}{2}$-uniformly sample from $\mathbb{F}_p$. Notice that for any $b \in \mathbb{F}_p$, $\frac{1-\epsilon/2}{p} \leq \text{Prob}_{x \in \mathbb{Z}_M}[x \equiv b \bmod p] \leq \frac{1+\epsilon/2}{p}$. Now using the Lemma 2.6.3, we conclude the following :

$$\begin{aligned}\text{Prob}_{a_1 \in U, a_2 \in U \cdots a_n \in U}[g(a_1, \cdots, a_n) = 0] &\leq \text{Prob}_{b_1 \in \mathbb{F}_p \cdots b_n \in \mathbb{F}_p}[\hat{g}(b_1, \cdots, b_n) = 0] \\ &\leq \frac{d}{p}(1 + \frac{\epsilon}{2}),\end{aligned}$$

where $b_i = a_i \bmod p$. The additional factor of $(1 + \frac{\epsilon}{2})$ comes from the fact that we are only sampling $\frac{\epsilon}{2}$-uniformly from $\mathbb{F}_p$. This can be easily verified from the proof of Lemma 2.6.3. Hence we have proved the lemma. ■

Again, the above result is only useful for large primes p. In particular, we get a constant success probability if $p \geq kd$ for a constant k.

2.7 Randomized Polynomial Identity Testing over Finite Rings

In this section we study the identity testing problem over finite commutative ring oracle with unity. For the input polynomial, we consider both black-box representation and circuit representation. First we consider the black-box case. Our identity testing algorithm is a direct consequence of Lemma 2.6.4.

Theorem 2.7.1 *Let R (which decomposes into local rings as $\oplus_{i=1}^{\ell} R_i$) be a finite commutative ring with unity given as a oracle. Let the input polynomial $f \in R[x_1, \cdots, x_n]$ of degree at most d be given via black-box access. Suppose R_i's is of characteristic $p_i^{\alpha_i}$. Let $\epsilon > 0$ be a given constant. If $p_i \geq kd$ for all i, for some integer $k \geq 2$, we have a randomized polynomial time identity test with success probability $1 - \frac{1}{k}(1 + \frac{\epsilon}{2})$.*

Proof. Consider the natural isomorphism $\hat{\phi} : R[x_1, x_2, \cdots, x_n] \to \oplus_{i=1}^{\ell} R_i[x_1, x_2, \cdots, x_n]$. Let $\hat{\phi}(f) = (f_1, f_2, \cdots, f_\ell)$. If $f \not\equiv 0$ then $f_i \not\equiv 0$ for some $i \in [\ell]$, where $f_i \in R_i[x_1, x_2, \cdots, x_n]$. Fix such an i. Our algorithm is a direct application of Lemma 2.6.4. Define $U = \{ce \mid 0 \leq c \leq M\}$, assign values for the x_i's independently and uniformly at random from U, and evaluate f using the black-box access. The algorithm declares $f \not\equiv 0$ if and only if the computed value is nonzero. By Lemma 2.6.4, our algorithm outputs the correct answer with probability $1 - \frac{d}{p_i}(1 + \frac{\epsilon}{2}) \geq 1 - \frac{1}{k}(1 + \frac{\epsilon}{2})$. [5] ∎

The drawback of Theorem 2.7.1 is that we get a randomized polynomial-time algorithm only when $p_i \geq kd$.

However, when the polynomial f is given by an arithmetic circuit we will get a randomized identity test that works for all finite commutative rings given by oracle. This is the main result in this section. A key idea is to apply the transformation from [AB03] to convert the given multivariate polynomial to a univariate polynomial. The following lemma has an identical proof as [AB03, Lemma 4.5].

[5] Notice that we have to compute ce using the ring oracle for addition in R. Starting with e, we need to add it c times. The running time for this computation can be made polynomial in $\log c$ by writing c in binary and applying the standard doubling algorithm.

Lemma 2.7.2 *Let R be an arbitrary commutative ring and $f \in R[x_1, x_2, \cdots, x_n]$ be any polynomial of maximum degree d. Consider the polynomial $g(x)$ obtained from $f(x_1, x_2, \cdots, x_n)$ by replacing x_i by $x^{(d+1)^{i-1}}$ i.e $g(x) = f(x, x^{(d+1)}, \cdots, x^{(d+1)^{n-1}})$. Then $f \equiv 0$ over $R[x_1, \cdots, x_n]$ if and only if $g \equiv 0$ over $R[x]$.*

By Lemma 2.7.2, it suffices to describe the identity test for a univariate polynomial in $R[x]$ given by an arithmetic circuit. Notice that if $\deg(f) = d$ then we can bound $\deg(g)$ by $d(d+1)^{n-1}$ which we denote by D. Our algorithm is simple and essentially the same as the Agrawal-Biswas identity test over the finite ring $\mathbb{Z}_n$ [AB03].

Recall the definition of $U[x]$ in the proof of the Theorem 2.7.1:

$$U = \{ce \mid 0 \leq c \leq M\}.$$

We will randomly pick a monic polynomial $q(x) \in U[x]$ of degree $\lceil \log O(D) \rceil$. Then we carry out a division of $f(x)$ by the polynomial $q(x)$ over the ring $R[x]$ and compute the remainder $r(x) \in R[x]$. Our algorithm declares f to be identically zero if and only if $r(x) = 0$. Notice that we will use the structure of the circuit to carry out the division. At each gate we carry out the division. More precisely, if the inputs of a $+$ gate are the remainders $r_1(x)$ and $r_2(x)$, then the output of this $+$ gate is $r_1 + r_2$. Similarly if r_1 and r_2 are the inputs of a $*$ gate, then we divide $r_1(x)r_2(x)$ by $q(x)$ and obtain the remainder as its output. Crucially, since $q(x)$ is a monic polynomial, the division algorithm will make sense and produce unique remainder even if $R[x]$ is not a U.F.D (which is the case in general).

The pseudocode of the identity testing algorithm is given in Algorithm 1. Our algorithm takes as input an arithmetic circuit C computing a polynomial $f \in R[x_1, x_2, \cdots, x_n]$ of degree at most d and an $\epsilon > 0$.

We will now prove the correctness of the above randomized identity test in Lemmas 2.7.3, 2.7.4, and 2.7.5.

Lemma 2.7.3 *Let R be a local commutative ring with unity and of characteristic p^α for some prime p and integer $\alpha > 0$. Let g be a nonzero polynomial in $R[x]$ such that $g \in p^k R[x] \setminus p^{k+1} R[x]$ for $k < \alpha$. Let $\hat{I} = \{r \in R \mid p^k r = 0\}$, $g = p^k \hat{g}$ where $\hat{g} \notin pR$ and q is a monic polynomial in $R[x]$. If q divides g in R, then q divides $\hat{g}$ in $R/(\hat{I} + pR)$.*

Proof. As $q(x)$ divides $g(x)$ in $R[x]$, we have $g(x) = q(x)q_1(x)$ for some polynomial $q_1(x) \in R[x]$. Suppose $\hat{g}(x) = q(x)\bar{q}(x) + r(x)$ in $R[x]$ where the degree of $r(x)$ is

Algorithm 1 The Identity Testing algorithm

```
1: procedure IdentityTesting($C$,$\epsilon$)
2:     for $i = 1, n$ do
3:         $x_i \leftarrow x^{(d+1)^{i-1}}$                          ▷ Univariate transformation
4:     end for
5:     $g(x) \leftarrow C(x, x^{(d+1)}, \cdots, x^{(d+1)^{n-1}})$.
6:     $D \leftarrow d(d+1)^{n-1}$.                ▷ The formal degree of $g(x)$ is at most $D$
7:     Choose a monic polynomial $q(x) \in U[x]$ of degree $\lceil \log \frac{12D}{1-\epsilon} \rceil$ uniformly at random.
8:     Divide $g(x)$ by $q(x)$ and compute the remainder $r(x)$. ▷ The division algorithm uses the structure of the circuit.
9:     if $r(x) = 0$ then
10:        $C$ computes a zero polynomial.
11:    else
12:        $C$ computes a nonzero polynomial.
13:    end if
14: end procedure
```

less than the degree of $q(x)$. Also note that the division makes sense even over the ring as $q(x)$ is monic. We want to show that $r(x) \in (\hat{I} + pR)[x]$. We have the following relation in $R[x]$:

$$g = qq_1 = p^k\hat{g} = p^kq\bar{q} + p^kr.$$

So, $p^kr = q(q_1 - p^k\bar{q})$. If $(q_1 - p^k\bar{q}) \not\equiv 0$ in $R[x]$, then the degree of the polynomial $q(q_1 - p^k\bar{q})$ is strictly more than the degree of p^kr as q is monic and degree of q is more than the degree of r. Thus $(qq_1 - p^kq\bar{q}) \equiv 0$ in $R[x]$ forcing $p^kr = 0$ in $R[x]$. So by the choice of $\hat{I}$, we have $r(x) \in \hat{I}[x]$. Thus $r(x) \in (\hat{I} + pR)[x]$. Notice that in Lemma 2.6.4, we have already proved that $\hat{g}(x) \not\equiv 0$ in $S[x]$, where $S = R/(\hat{I} + pR)$. Also q is nonzero in $S[x]$ as it is a monic polynomial. Hence we have proved that $q(x)$ divides $\hat{g}(x)$ over $S[x]$. ■

The following lemma is basically Chinese remaindering tailored to our setting.

Lemma 2.7.4 *Let R be a local ring with characteristic p^α. Let $g(x) \in p^kR[x] \backslash p^{k+1}R[x]$ for some $k \geq 0$. Let $g(x) = p^k\hat{g}(x)$ and $\hat{I} = \{r \in R \mid p^kr = 0\}$. Suppose $q_1(x), q_2(x)$ are two monic polynomials over $R[x]$ such that each of them divides g in $R[x]$. Moreover, suppose there exist polynomials $a(x), b(x) \in R[x]$ such that $aq_1 + bq_2 = 1$ in $R/(\hat{I} + pR)$.*

Then q_1q_2 divides $\hat{g}$ in $R/(\hat{I}+pR)$.

Proof. By Lemma 2.7.3, we know that q_1 and q_2 divide $\hat{g}$ in $R/(\hat{I}+pR)$. Let $\hat{g} = q_1\bar{q}_1$ and $\hat{g} = q_2\bar{q}_2$ in $R/(\hat{I}+pR)$. Let $\bar{q}_1 = q_2q_3 + r$ in $R/(\hat{I}+pR)$. So, $\hat{g} = q_1q_2q_3 + q_1r$. Substituting $q_2\bar{q}_2$ for $\hat{g}$, we get $q_2(\bar{q}_2 - q_1q_3) = q_1r$. Multiplying both side by a and substituting $aq_1(x) = 1 - bq_2$, we get $q_2[a(\bar{q}_2 - q_1q_3) + br] = r$. If $r \not\equiv 0$ in $R/(\hat{I}+pR)$, we arrive at a contradiction since q_2 is monic and thus the degree of $q_2[a(\bar{q}_2 - q_1q_3) + br]$ is more than the degree of r. ■

Let $f(x)$ be a nonzero polynomial in $R[x]$ of degree at most D. The next lemma states that, if we pick a random monic polynomial $q(x) \in U[x]$ (recall that $U = \{ce \mid 0 \leq c \leq M\}$) of degree $d \approx \log O(D)$, with high probability, $q(x)$ will not divide $f(x)$.

Lemma 2.7.5 *Let R be a commutative ring with unity. Suppose $f(x) \in R[x]$ is a nonzero polynomial of degree at most D and $\epsilon > 0$ be a given constant. Choose a random monic polynomial $q(x)$ of degree $d = \lceil \log \frac{12D}{1-\epsilon} \rceil$ in $U[x]$. Then with probability at least $\frac{1-\epsilon}{4d}$, $q(x)$ will not divide $f(x)$ over $R[x]$.*

Proof. Let $R \cong \bigoplus_i R_i$ is the local ring decomposition of R. As f is nonzero in $R[x]$, there exists j such that $f_j = \hat{\phi}_j(f)$ is nonzero in $R_j[x]$. Clearly, we can lower bound the required probability by the probability that $q_j = \hat{\phi}_j(q)$ does not divide f_j in $R_j[x]$. Let the characteristic of R_j is p^α. If q_j divides f_j in $R_j[x]$, then it also divides over $R_j/(\hat{I}_j + pR_j)$. It is shown in the proof of Lemma 2.6.4, $\mathbb{F}_p \subset R_j/(\hat{I}_j + pR_j)$. Now the number of irreducible polynomials in $\mathbb{F}_p$ of degree d is at least $\frac{p^d - 2p^{d/2}}{d}$. Let $t = \frac{p^d - 2p^{d/2}}{d}$. Let $\hat{q}(x) = \sum_{i=0}^{d-1} b_ix^i + x^d \in \mathbb{F}_p[x]$ be a monic polynomial. Now if a monic polynomial $P(x)$ of degree d is randomly chosen from $U[x]$ then, Prob$[P(x) \equiv \hat{q}(x) \bmod p] = \frac{\prod_{i=0}^{d-1} \lfloor (M-b_i)/p \rfloor + 1}{M^d} \geq \frac{1}{p^d}(1 - \frac{2^m}{M})^d$. Again, choosing $M > d2^{m+1}/\epsilon$, we get Prob$[P(x) \equiv \hat{q}(x) \bmod p] \geq (1 - \epsilon/2)/p^d$.

So, the probability that q_j is an irreducible polynomial in $\mathbb{F}_p[x]$ is at least $t(1 - \epsilon)/p^d > (1-\epsilon)/2d$. Let $f_j \in p^kR_j[x] \setminus p^{k+1}R_j[x]$. So we can write $f_j = p^kf'$, where $f' \in R_j[x] \setminus pR_j[x]$. By the Lemma 2.7.3, q_j divides f' in $R/(\hat{I}_j + pR)$. Also, by Lemma 2.7.4, the number of different monic polynomials that are irreducible in $\mathbb{F}_p$ and divides f' in $R_j/(\hat{I}_j + pR_j)$ is at most D/d. In the sample space for q, any monic polynomial of degree d in $R_j/(\hat{I}_j + pR_j)$ occurs at most $(\frac{M}{p} + 1)^d$ times. So the probability that a random monic irreducible polynomial q will divide f is at most

$\frac{(D/d)(\frac{M}{p}+1)^d}{M^d} \leq \frac{D}{dp^d}(1+\frac{1}{d})^d < \frac{3D}{d2^d}$. So a random monic polynomial $q \in U[x]$ (which is irreducible in $\mathbb{F}_p$ with reasonable probability) will not divide $f(x)$ with probability at least $\frac{1-\epsilon}{2d} - \frac{3D}{dp^d} > \frac{1-\epsilon}{4d}$ for $d \geq \lceil \log \frac{12D}{1-\epsilon} \rceil$. ■

The correctness of Algorithm 1 and its success probability follow directly from Lemma 2.7.3, Lemma 2.7.4 and Lemma 2.7.5.

In particular, by Lemma 2.7.5, the success probability of our algorithm is at least $\frac{1-\epsilon}{4t}$, where $t = \lceil \log \frac{12D}{1-\epsilon} \rceil$. As $\frac{1-\epsilon}{4t}$ is an inverse polynomial quantity in input size and the randomized algorithm has one-sided error, we can boost the success probability by repeating the test polynomially many times. We summarise the result in the following theorem.

Theorem 2.7.6 *Let R be a finite commutative ring with unity given as an oracle and $f \in R[x]$ be a polynomial, given as an arithmetic circuit. Then in randomized time polynomial in the circuit size and* $\log\ |R|$ *we can test whether $f \equiv 0$ in $R[x]$.*

Randomized polynomial-time identity testing for $f \in R[x_1, \cdots, x_n]$ given by arithmetic circuits, follows from Theorem 2.7.6 and Lemma 2.7.2.

Theorem 2.7.7 *Let R be a commutative ring with unity given as an oracle. Let f be a polynomial in $R[x_1, x_2, \cdots, x_n]$ of formal degree at most d, is given by an arithmetic circuit over R. Then in randomized time polynomial in circuit size and* $\log |R|$ *we can test whether $f \equiv 0$ in $R[x_1, x_2, \cdots, x_n]$.*

3 Ideal Membership and Polynomial Identity Testing

3.1 Introduction

For a field $\mathbb{F}$ let $\mathbb{F}[x_1, x_2, \cdots, x_n]$ be the ring of polynomials over $\mathbb{F}$ with indeterminates $x_1, x_2, \cdots, x_n$. Let $I \subseteq \mathbb{F}[x_1, x_2, \cdots, x_n]$ be an ideal given by a finite generator set $\{g_1, g_2, \cdots, g_r\}$ of polynomials. Then $I = \{\sum_{i=1}^{r} a_i g_i \mid a_i \in \mathbb{F}[x_1, x_2, \cdots, x_n]\}$, and we write $I = \langle g_1, g_2, \cdots, g_r \rangle$.

Given an ideal $I = \langle g_1, g_2, \cdots, g_r \rangle$ and a polynomial $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ the *Ideal Membership* problem is to decide if $f \in I$.

Ideal Membership Testing is a fundamental algorithmic problem with important applications [CLO92]. In general, however, Ideal Membership Testing is highly intractable. The results of Mayr and Meyer show that it is EXPSPACE-complete [MM82, May89]. Nevertheless, because of its important applications, algorithms for this problem are widely studied, mainly based on the theory of Gröbner bases (see [CLO92] and [vzGG03]).

In this chapter we study interesting connections between Ideal Membership problem and Polynomial Identity Testing. In particular we will study the connection between Monomial Ideal Membership and Polynomial Identity Testing. The study of monomial ideals is central to the theory of Gröbner bases [CLO92]. In Section 3.2 we explain this in more detail.

Suppose $I = \langle m_1, m_2, \cdots, m_k \rangle$ is a monomial ideal in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ generated by the monomials m_i. In contrast to the general ideal membership problem, testing membership in the monomial ideal I is trivial for a polynomial $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ that is given explicitly as an $\mathbb{F}$-linear combination of monomials. We only need to check if each monomial occurring in f is divisible by some generator monomial m_i. However, as we show in this chapter, the problem becomes interesting when f is given by an arithmetic circuit. In that case, it turns out that the problem is tractable when k is a constant and its complexity is similar to that of polynomial identity testing. Given a monomial ideal $I = \langle m_1, m_2, \cdots, m_r \rangle$ for monomials $m_i \in \mathbb{F}[x_1, \cdots, x_n]$ and an arithmetic circuit C over $\mathbb{F}$ defining a polynomial $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$, the *Monomial Ideal Membership* problem is to decide if $f \in I$. Note that, whenever there is an ideal given by a generating set of monomials or polynomials (in general), we will always assume that the exponent of any variable that appears in a generator, is given in unary.

We study different versions of the problem by placing restrictions on the arithmetic circuit C and the number of monomials generating the ideal I. We also consider a more general version of the problem where we are allowed only black-box access to the polynomial f. The main results of this chapter are as follows.

We show a randomized test for Monomial Ideal Membership when f is given by an arithmetic circuit and $I = \langle m_1, m_2, \cdots, m_k \rangle$ for constant k. This is analogous to the Schwartz-Zippel randomized polynomial identity test [Sch80, Zip79]. When k is unrestricted, we show that the problem is coNP-hard, but we are able to show an upper bound in the counting hierarchy.

The identity testing problem for $\Sigma\Pi\Sigma$ circuits has recently attracted a lot of research [DS06, KS07, KS08]. The main open problem is whether there is a deterministic polynomial-time identity test for $\Sigma\Pi\Sigma$ circuits. For the special case of $\Sigma\Pi\Sigma$ circuits with bounded fan-in output gate Kayal and Saxena [KS07] recently gave an ingenious deterministic polynomial-time test. Analogous to their result, we consider monomial ideal membership, where f is computed by a $\Sigma\Pi\Sigma$ circuit with bounded fan-in output gate, and $I = \langle m_1, m_2, \cdots, m_k \rangle$ for constant k. Using the algorithm of [KS07] we give a *deterministic* polynomial-time algorithm for this Monomial Ideal Membership problem. More interestingly, we develop the algorithm and its correctness proof based on Gröbner basis theory. We believe this approach is somewhat simpler and direct as compared to [KS07]. It avoids properties such as Chinese remaindering in local rings and Hensel lifting that is used in [KS07]. As a byproduct, this gives us a different understanding of the identity testing algorithm of [KS07].

3.2 Preliminaries

We develop the rudiments of Gröbner basis theory. Details can be found in the texts [CLO92, vzGG03] and Madhu Sudan's notes [Sud98].

Let $\bar{x}$ denotes indeterminates $\{x_1, x_2, \cdots, x_n\}$. Let $\mathbb{F}[\bar{x}]$ denotes the polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$. For a commutative ring R, a subring $I \subseteq R$ is an *ideal* of R if $IR \subseteq R$. The Hilbert basis theorem [CLO92, Theorem 4, pp.74] states that any ideal I of $\mathbb{F}[x_1, x_2, \cdots, x_n]$ is *finitely generated*. I.e. we can express $I = \{\sum_{i=1}^{r} p_i g_i \mid p_i \in \mathbb{F}[x_1, x_2, \cdots, x_n]\}$, where the finite collection of polynomials $\{g_1, g_2, \cdots, g_r\}$ is a generating set (or basis) for I.

The notion of monomial ordering is key to defining Gröbner bases. We restrict ourselves to the *lexicographic monomial ordering* which we define below. For $\bar{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathbb{N}^n$, we denote the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ by $\bar{x}^{\bar{\alpha}}$.

Definition 3.2.1 *Let $\bar{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ and $\bar{\beta} = (\beta_1, \beta_2, \cdots, \beta_n) \in \mathbb{N}^n$. We say $\bar{\alpha} > \bar{\beta}$ if, in the vector difference $\bar{\alpha} - \bar{\beta} \in \mathbb{N}^n$, the left-most nonzero entry is positive. We say, $\bar{x}^{\bar{\alpha}} > \bar{x}^{\bar{\beta}}$ (equivalently, $\bar{x}^{\bar{\beta}} < \bar{x}^{\bar{\alpha}}$) if $\bar{\alpha} > \bar{\beta}$.*

The lexicographic monomial ordering naturally fixes a leading monomial $LM(f)$ for any polynomial f. Let $LC(f)$ denotes the coefficient of $LM(f)$. Then the *leading term* of f is $LT(f) = LC(f)LM(f)$. Using the monomial ordering, we state the general form of the division algorithm over $\mathbb{F}[x_1, x_2, \cdots, x_n]$.

Theorem 3.2.2 [CLO92, Theorem 3, pp.61] *Let $f \in \mathbb{F}[\bar{x}]$ and $(f_1, f_2, \cdots, f_s)$ be an ordered s-tuple of polynomials in $\mathbb{F}[\bar{x}]$. Then f can be written as, $f = a_1 f_1 + a_2 f_2 + \cdots + a_s f_s + r$, where $a_i, r \in \mathbb{F}[\bar{x}]$, and either $r = 0$ or r is an $\mathbb{F}$-linear combination of monomials, none of which is divisible by any of $LT(f_1), LT(f_2), \cdots, LT(f_s)$.*

The proof of the theorem is constructive. We give an intuitive outline of the proof. Let $\bar{f}$ denotes the ordering of the polynomials f_i's: $\bar{f} = (f_1, f_2, \cdots, f_s)$. The proof describes a division algorithm Divide$(f; \bar{f})$ which first sorts f by the monomial ordering. The algorithm proceeds iteratively. It tries to eliminate the leading monomial in the current remainder by attempting to divide it with the f_i's in the given order. The f_i that succeeds is the first one whose leading monomial divides the leading monomial of the current remainder. Finally, the remainder r that survives has the above property. The algorithm is guaranteed termination as the monomial ordering is a well ordering. The following time bound for Divide$(f;\bar{f})$ is easy to obtain.

Fact 3.2.3 [Sud98, Section 6, pp.12-5] *The running time of Divide$(f;\bar{f})$ is bounded by $O(s\prod_{i=1}^{n}(d_i+1)^{O(1)})$, where d_i is the maximum degree of x_i among the polynomials $f, f_1, f_2, \cdots, f_s$.*

If the remainder r output by Divide$(f;\bar{f})$ is zero then clearly $f \in \langle f_1, \cdots, f_s\rangle$. However, in general, Divide$(f;\bar{f})$ need not produce zero remainder even if $f \in \langle f_1, \cdots, f_s\rangle$ as the order of division is important. Thus, it cannot be directly used as an ideal membership test. In order to ensure this property, we define *Gröbner bases* (with respect to the lexicographic monomial ordering).

Definition 3.2.4 *Fix $<$ as the monomial ordering, and let $J \subseteq \mathbb{F}[\bar{x}]$ be any ideal. Then the polynomials $g_1, g_2, \cdots, g_t$ form a* Gröbner basis *for J if $J = \langle g_1, g_2, \cdots, g_s\rangle$ and $\langle LT(g_1), \cdots, LT(g_t)\rangle = \langle LT(J)\rangle$, where $\langle LT(J)\rangle$ is the ideal generated by the leading terms of the polynomials in J.*

The following lemma states that the general division algorithm of Theorem 3.2.2 carried out w.r.t. a Gröbner basis results in a unique remainder r regardless of the order in which division is applied.

Lemma 3.2.5 [Proposition 1, pp.79][CLO92] Let $G = \{f_1, f_2, \cdots, f_s\}$ be a Gröbner basis for an ideal $J \subseteq \mathbb{F}[\bar{x}]$ and $f \in \mathbb{F}[\bar{x}]$. Then there is a *unique* polynomial $r \in \mathbb{F}[\bar{x}]$ such that f can be written as, $f = a_1f_1 + a_2f_2 + \cdots + a_sf_s + r$, for $a_i \in \mathbb{F}[\bar{x}]$, and either $r = 0$ or r is an $\mathbb{F}$-linear combination of monomials, none of which is divisible by any of $LT(f_1), LT(f_2), \cdots, LT(f_s)$. In particular, for every ordering $\hat{g}$ of G, r is the unique remainder when Divide$(f, \hat{g})$ is invoked.

By Lemma 3.2.5, for an ideal J and a polynomial f, we can indeed test if $f \in J$ given a Gröbner basis $\bar{f} = \{f_1, f_2, \cdots, f_s\}$ for J. We need to compute Divide$(f;\bar{f})$ and check if the remainder is zero.

The following theorem gives us an easy to test sufficient condition to check if a given generating set for an ideal is already a Gröbner basis.

Theorem 3.2.6 [CLO92, Theorem 3, Proposition 4, pp.101] *Let I be a polynomial ideal given by a basis $G = \{g_1, g_2, \cdots, g_s\}$ such that all pairs $i \neq j$ $LM(g_i)$ and $LM(g_j)$ are relatively prime. Then G is a Gröbner basis for I.*

Recall from the introduction that a *monomial ideal* is an ideal generated by a finite set of monomials in $\mathbb{F}[\bar{x}]$. Indeed, by Dickson's Lemma, an ideal generated by an arbitrary subset of monomials is also generated by a finite subset of monomials and hence is a monomial ideal. We recall the formal statement of Dickson's Lemma.

Lemma 3.2.7 *(Dickson's Lemma)[CLO92, Theorem 5, pp.69] Any monomial ideal $I = \langle \bar{x}^{\bar{e}} \mid \bar{e}\mathbb{Z}^n \rangle$ in $\mathbb{F}[\bar{x}]$ can be written down in the form $I = \langle \bar{x}^{\bar{\alpha_1}}, \bar{x}^{\bar{\alpha_2}}, \cdots, \bar{x}^{\bar{\alpha_s}} \rangle$ for some s. In particular, I has a finite monomial basis.*

An interesting property of monomial ideals is the following.

Lemma 3.2.8 [CLO92, Lemma 2, Lemma 3, pp.67-68] *Let $I = \langle m_1, m_2, \cdots, m_s \rangle$ be a monomial ideal and $f \in \mathbb{F}[\bar{x}]$. Then $f \in I$ if and only if each monomial of f is in I. Furthermore, a monomial m is in the ideal I if and only if there exist $i \in [s]$, such that m_i divides m.*

An immediate consequence of Lemma 3.2.8 is that we can test in deterministic polynomial time if an explicitly given polynomial $f \in \mathbb{F}[\bar{x}]$ is in a monomial ideal I.

3.3 Monomial Ideal Membership

In this section we consider monomial ideal membership when f is given by an arithmetic circuit. We show that the problem can be solved in randomized polynomial time if number of generators k for the monomial ideal I is a constant. When k is not a constant we show that it is coNP-hard and is contained in coAM$^{\text{PP}}$. We leave open a tight classification of the complexity of this problem.

Lemma 3.3.1 *Let, $I = \langle m_1, m_2, \cdots, m_k \rangle$ be a monomial ideal in $\mathbb{F}[x_1, x_2, \cdots, x_n]$. For $i \in [k]$, let $m_i = x_1^{e_{i1}} x_2^{e_{i2}} \cdots x_n^{e_{in}}$. Let $\bar{v}$ be a k-tuple given by $\bar{v} = (j_1, j_2, \cdots, j_k)$, where $j_i \in [n]$. Define the ideal, $I_{\bar{v}} = \langle x_{j_1}^{e_{1j_1}}, \cdots, x_{j_k}^{e_{kj_k}} \rangle$. Then $f \in I$ if and only if, $\forall \bar{v} \in [n]^k$, $f \in I_{\bar{v}}$.*

Proof. Let $f \in I$. So f can be written as $f = p_1 m_1 + u_2 m_2 + \cdots + p_k m_k$, where $p_i \in \mathbb{F}[\bar{x}]$ for all i. Then clearly $\forall \bar{v} \in [n]^k$, $f \in I_{\bar{v}}$. To see the other direction, suppose $f \notin I$. Write $f = c_1 M_1 + c_2 M_2 + \cdots + c_t M_t$, where M_i's are the monomials of f and $c_i \in \mathbb{F}$ are the corresponding coefficients. As $f \notin I$, there is a $j \in [t]$, such that $M_j \notin I$. Thus, for all $i \in [k]$, m_i does not divide M_j. So each of the m_i's contains some x_{ℓ_i} such that the exponent of x_{ℓ_i} is greater than the exponent of x_{ℓ_i} in M_j. Let $\{\ell_1, \ell_2, \cdots, \ell_k\}$ be k such indexes. Now consider the ideal $I_{\bar{w}}$, where $\bar{w} = (\ell_1, \ell_2, \cdots, \ell_k)$. By Lemma 3.2.8, $M_j \notin I_{\bar{w}}$ and hence $f \notin I_{\bar{w}}$. ∎

Using Lemma 3.3.1, we generalize the Schwartz-Zippel Lemma for Polynomial Identity Testing to a form tailored for Monomial Ideal Membership.

Lemma 3.3.2 *Let $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ be a polynomial of total degree d and $I = \langle x_1^{e_1}, x_2^{e_2}, \cdots, x_k^{e_k} \rangle$ be a monomial ideal (as described in Lemma 3.3.1). Fix a finite subset $S \subseteq \mathbb{F}$, and let $r_1, r_2, \cdots, r_{n-k}$ be chosen independently and uniformly at random from S. Then,*

$$\text{Prob}_{r_i \in S}[f(x_1, x_2, \cdots, x_k, r_1, r_2, \cdots, r_{n-k}) \in I \mid f \notin I] \leq \frac{d}{|S|}.$$

Proof. First we write $f = \sum_{\bar{v}} x_1^{j_1} \cdots x_k^{j_k} f_{\bar{v}}(x_{k+1}, \cdots, x_n)$, where $\bar{v} = (j_1, \cdots, j_k)$. Any term in the above expression with $j_i \geq e_i$ is already in I. Thus, it suffices to consider the sum $\hat{f}$ of the remaining terms. More precisely, Let $\mathcal{A} = [e_1 - 1] \times [e_2 - 1] \times \cdots \times [e_k - 1]$. We can write $\hat{f} = \sum_{\bar{v} \in \mathcal{A}} x_1^{j_1} \cdots x_k^{j_k} f_{\bar{v}}(x_{k+1}, \cdots, x_n)$ where $\bar{v} = (j_1, j_2, \cdots, j_k) \in \mathcal{A}$. As $\hat{f} \notin I$, not all $f_{\bar{v}}$ are identically zero. Choose and fix one such $\bar{u}$. By the Schwartz-Zippel lemma [Sch80, Zip79],

$$\text{Prob}_{r_i \in S}[f_{\bar{u}}(r_1, r_2, \cdots, r_{n-k}) = 0 \ \mid \ f_{\bar{u}}(x_{k+1}, x_{k+2}, \cdots, x_n) \not\equiv 0] \leq \frac{d}{|S|}.$$

Notice that for any $\bar{v} = (j_1, j_2, \cdots, j_k) \in \mathcal{A}$, the monomial $x_1^{j_1} \cdots x_k^{j_k}$ is not in I. Thus,

$$f(x_1, x_2, \cdots, x_k, r_1, r_2, \cdots, r_{n-k}) \in I$$

if and only if,

$$\forall \bar{v}, f_{\bar{v}}(r_1, r_2, \cdots, r_{n-k}) = 0.$$

But $f_{\bar{u}}(r_1, r_2, \cdots, r_{n-k}) = 0$ with probability at most $d/|S|$. This completes the proof. ∎

Now using Lemma 3.3.2, we prove the following theorem.

Theorem 3.3.3 *Let $f \in \mathbb{F}[\bar{x}]$ be given by an arithmetic circuit C of size s and the ideal $I = \langle m_1, m_2, \cdots, m_k \rangle$ generated by monomials m_i's where k is a constant. For such instances Monomial Ideal Membership can be solved in randomized polynomial time (in $n^{O(k)}$ time).*

Proof. First, we construct all the ideals, $\{I_{\bar{v}} \mid \bar{v} \in [n]^k\}$ as described in Lemma 3.3.1. Then for each such $I_{\bar{v}}$, we check if $f \in I_{\bar{v}}$. The correctness of the algorithm follows from Lemma 3.3.1. Let $I_{\bar{v}} = \langle x_1^{e_1}, x_2^{e_2}, \cdots, x_k^{e_k} \rangle$. To check $f \in I_{\bar{v}}$, we assign random values to $x_{k+1}, \cdots, x_n$ from S and then evaluate the circuit C in the ring $R = \mathbb{F}[x_1, x_2, \cdots, x_k]/I_{\bar{v}}$. To evaluate the circuit in R, we need to compute each gate operation modulo $I_{\bar{v}}$, starting from the input gates. Notice that, as $\langle x_1^{e_1}, x_2^{e_2} \cdots, x_k^{e_k} \rangle$ is a Gröbner basis for $I_{\bar{v}}$, by Lemma 3.2.5 the actual order in which we evaluate the gates is not important. Let, $e = \sum_{i=1}^{k} e_i$. Then it is easy to see that the running time of the algorithm is poly(n, s, e^k) (notice that e_i's are in unary). Furthermore, by Lemma 3.3.2, the success probability of the algorithm is seen to be $\geq 1 - (d/|S|)$ where d is the total degree of f. Thus it is enough to consider sampling from a set S s.t, $|S| = 2d$ using $O(n \log d)$ random bits which is polynomial in the input size. ■

When the monomial ideal I is not generated by a constant number of monomials the monomial ideal membership problem is coNP hard over any field.

Theorem 3.3.4 *Given a polynomial f as an arithmetic circuit, and a monomial ideal $I = \langle m_1, m_2, \cdots, m_k \rangle$ (k is not a constant), it is* coNP*-hard to test whether $f \in I$.*

Proof. Indeed, we prove the coNP-hardness even for f given by a $\Pi\Sigma$ arithmetic circuit. First we consider the case when the field $\mathbb{F}$ is $\mathbb{Q}$. We give a reduction from 3-CNF. Let $F = C_1 \wedge C_2 \wedge \cdots \wedge C_\ell$ is a 3-CNF formula over $\{x_1, x_2, \cdots, x_n\}$, with C_i are the clauses. Introduce new variables $\{y_1, y_2, \cdots, y_n\}$ for $\{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n\}$. Next, we encode each of the clause as a linear form (sum of variables). For example, if $C_1 = x_1 \vee x_2 \vee \bar{x}_3$ then we encode it as $x_1 + x_2 + y_3$. Thus we get a polynomial C corresponding to F : $C(\bar{x}, \bar{y}) = \prod_{i=1}^{\ell} L_i(\bar{x}, \bar{y})$, where L_i's are the linear form corresponding to C_i. Clearly, $C(\bar{x}, \bar{y})$ represents a $\Pi\Sigma$ circuit. Define a monomial ideal, $I = \langle x_i y_i \mid 1 \leq i \leq n \rangle$. It follows that, if F is satisfiable then not all the monomials of C are in I. In that case $C \notin I$ by Lemma 3.2.8. Conversely assume that $C \notin I$. That means, C has at least one monomial m such that m does not contain both x_i and y_i for any i. Thus, the variables of m correspond to a satisfying assignment for F (set the variables those are not in m to zero).

Now, let the characteristic of the field be finite. The only place the proof differs from the above is, we need to encode each clause as a sum of all seven monomials representing the satisfying assignment of that clause. For example, an assignment $\{1, 0, 1\}$ of $\{x_1, x_2, x_3\}$ corresponds to a monomial $x_1y_2x_3$. Thus a clause $C_1 = x_1 \vee x_2 \vee \bar{x}_3$ will be encoded as a sum of all possible monomials except $y_1y_2x_3$. Note that the polynomial C corresponding to F is represented by a $\Pi\Sigma\Pi$ circuit. The rest of the argument follows exactly as above. ■

Next, we show an upper bound for Monomial Ideal Membership when the number of monomial generators is not restricted to a constant.

Theorem 3.3.5 *For $\mathbb{F} = \mathbb{Q}$, Monomial Ideal Membership is in* coAM$^{\text{PP}}$ *where the input monomial ideal $I = \langle m_1, m_2, \cdots, m_k\rangle$ is given by a list of monomials and $f \in \mathbb{F}[\bar{x}]$ is given by an arithmetic circuit C. For $\mathbb{F} = \mathbb{F}_p$, Monomial Ideal Membership is in* coNP$^{\text{Mod}_p\text{P}}$.

Proof. For the first part, $\mathbb{F} = \mathbb{Q}$ and let C be the input arithmetic circuit computing $f \in \mathbb{F}[\bar{x}]$ and the monomial ideal I is $\langle m_1, m_2, \cdots, m_k\rangle$. We'll show that *Nonmembership* is in AM$^{\text{PP}}$. It suffices for the AM$^{\text{PP}}$ algorithm to exhibit a nonzero monomial m of f such that $m \notin \langle m_1, m_2, \cdots, m_k\rangle$. I.e. m_i does not divide m for $i = 1, 2, \cdots, k$. The base AM machine (call it M) will guess such a monomial $m = x_1^{e_1}x_2^{e_2}\cdots x_n^{e_n}$ by nondeterministically picking the tuple $(e_1, \cdots, e_n) \in \mathbb{N}^n$ and check that m_i does not divide m for all i. It remains to verify that m is a nonzero monomial of f. W.l.o.g. we can assume that $f \in \mathbb{Z}[\bar{x}]$. We will describe a BPP$^{\#\text{P}}$ algorithm that takes as input $\langle C, m\rangle$ and makes one #P query to decide if m is a nonzero monomial in f.

Write f as a finite sum $f = \sum_{\bar{\alpha}\in\mathbb{N}^n} c_\alpha \bar{x}^{\bar{\alpha}}$. Since the input to C are the indeterminates and constants, the numbers $c_{\bar{\alpha}}$ are bounded in absolute value by 2^K, where the size of $K \in \mathbb{Z}^+$ in binary is bounded by some polynomial in input size. Now, we observe that $c_{\bar{e}} \neq 0$ if and only if m occurs in f, where $\bar{e} = (e_1, e_2, \cdots, e_n)$. The BPP machine guesses a random prime p of polynomial size, where the size is chosen suitably, so that $c_{\bar{e}} \neq 0$ if and only if $c_{\bar{e}} \neq 0 \bmod p$ with high probability. Now we define the #P query that the BPP machine will make by defining a suitable NP machine N. The input to N is the triple (m, C, p) and the number of accepting paths has the property

$acc_N(m, C, p) = c_{\bar{e}} \bmod p$. Such an NP machine N would clearly suffice. We now define the NP machine N.

W.l.o.g. we can assume that each gate of C has fan-in two and is either a multiply gate or a plus gate. Suppose there are t plus gates in C. The NP machine N nondeterministically branches into 2^t computation paths, where on each path it picks exactly one of the two inputs to the plus gate. As a result, on each of the 2^t computation paths N has picked a multiplicative subcircuit of C. Let $\pi \in \{0, 1\}^t$ denote such a computation path of N and let C_π denote the corresponding multiplicative subcircuit of C. Notice that each C_π defines a monomial with a coefficient $c_\pi m_\pi$, and from C_π in deterministic polynomial time we can compute m_π and $c_\pi \bmod p$. Next, machine N proceeds as follows: if $m_\pi = m$ then N extends π into $c_\pi \bmod p$ accepting computation paths, and otherwise N rejects along π. Clearly, $acc_N(m, C, p) = c_{\bar{e}} \bmod p$.

For the second part when $\mathbb{F} = \mathbb{F}_p$ the proof is similar. The crucial difference is that we do not need to evaluate the circuit modulo a randomly chosen prime. Furthermore, we only need the number of accepting paths of N modulo p. Hence a Mod$_p$P oracle suffices with an NP base machine. ∎

3.4 Monomial Ideal Membership for $\Sigma\Pi\Sigma$ circuits

Consider instances (f, I) of Monomial Ideal Membership where f is given by a $\Sigma\Pi\Sigma$ circuit with top gate of bounded fan-in and $I = \langle m_1, m_2, \cdots, m_k \rangle$ a monomial ideal for constant k. By Lemma 3.3.1 this problem reduces to testing if f is in a monomial ideal of the form $I = \langle x_1^{e_1}, x_2^{e_2}, \cdots, x_k^{e_k} \rangle$. As the quotient ring $\mathbb{F}[x_1, x_2, \cdots, x_k]/I$ is a local ring and $f \in I$ if and only if $f \equiv 0$ over the local ring $\mathbb{F}[x_1, x_2, \cdots, x_k]/I$ we can apply the Kayal-Saxena deterministic identity test [KS07] for such $\Sigma\Pi\Sigma$ circuit over local rings[1] to check this in overall time polynomial in the circuit size.

However, in this section we develop the algorithm and its correctness proof based on Gröbner basis theory. The algorithm is essentially from [KS07]. But the Gröbner basis approach is somewhat simpler and direct. It avoids invoking properties such as Chinese remaindering in local rings and Hensel lifting. The added bonus is that we get a different correctness proof for the Kayal-Saxena identity test.

[1]More precisely, over local rings that allow polynomial-time arithmetic in them.

Definition 3.4.1 *A $\Sigma\Pi\Sigma$ circuit C with n inputs over a field $\mathbb{F}$ computes a polynomial of the form: $C(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{\ell} \prod_{j=1}^{d_i} L_{ij}(x_1, x_2, \cdots, x_n)$, where ℓ is the fan-in of the top Σ gate, and d_i are the fan-ins of the k different Π gates and L_{ij}'s are linear forms over $\mathbb{F}[x_1, x_2, \cdots, x_n]$.*

First, we transform the circuit C into another circuit C' as follows: Let $L_{ij} = \sum_{t=1}^{n} \alpha_{ijt}x_t + \beta$ for $\alpha_{ijt}, \beta \in \mathbb{F}$. We replace each such L_{ij} by $L'_{ij} = \sum_{t=1}^{n} \alpha_{ijt}x_t + \beta y$, where y is a new indeterminate. Let d be the maximum of the fan-ins of the Π gates. For a Π gate of fan-in d_i introduce $d - d_i$ new input fan-in wires each carrying y.

Proposition 3.4.2 *For $I = \langle x_1^{e_1}, x_2^{e_2}, \cdots, x_k^{e_k} \rangle$ and a $\Sigma\Pi\Sigma$ circuit C defined as above, $C \in I$ if and only if $C' \in \langle x_1^{e_1}, x_2^{e_2}, \cdots, x_k^{e_k}, y - 1 \rangle$.*

Notice that in the process of making this transformation the resulting ideal is not a monomial ideal any more.

Thus, we can assume that in the circuit C itself every L_{ij} is of the form $\sum_{t=1}^{n} \alpha_t x_t$ and the degree of the polynomial computed at each Π gate is d. We can naturally associate to L_{ij} its coefficient vector $(\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathbb{F}^n$. A collection of linear forms is *independent* if their coefficient vectors forms a linearly independent set in $\mathbb{F}^n$.

First we fix some notation. Let R denote the polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_k]$, where k will be clear from the context where R is used. For $\alpha = (e_{k+1}, e_{k+2}, \cdots, e_n) \in \mathbb{N}^{n-k}$, let $\bar{x}^{\bar{\alpha}}$ denote $x_{k+1}^{e_{k+1}} x_{k+2}^{e_{k+2}} \cdots x_n^{e_n}$. The only monomial ordering we use is the lex-ordering defined in Definition 3.2.1 w.r.t. the order $x_1 < x_2 < \cdots < x_n$. We can consider an $f \in \mathbb{F}[x_1, \cdots, x_n]$ as a polynomial in $R[x_{k+1}, x_{k+2}, \cdots, x_n]$. More precisely, we can write $f = \sum_{\bar{\alpha} \in \mathbb{N}^{n-k}} A_{\bar{\alpha}} \bar{x}^{\bar{\alpha}}$, where $A_{\bar{\alpha}} \in \mathbb{F}[x_1, x_2, \cdots, x_k] \setminus \{0\}$. Let $\bar{\alpha_1}$ be such that $\bar{x}^{\bar{\alpha_1}}$ is the lex-largest term such that $A_{\bar{\alpha_1}} \neq 0$. Then we denote the R-leading term $A_{\bar{\alpha_1}} \bar{x}^{\bar{\alpha_1}}$ of f by $LT_R(f)$. Likewise, $LM_R(f) = \bar{x}^{\bar{\alpha_1}}$ and $LC_R(f) = A_{\bar{\alpha_1}}$ is the R-leading monomial and R-leading coefficient of f. For any $f, g \in \mathbb{F}[x_1, \cdots, x_n]$, it is clear that $LM_R(fg) = LM_R(f)LM_R(g)$, $LC_R(fg) = LC_R(f)LC_R(g)$.

Let $f \in \mathbb{F}[x_1, \cdots, x_n]$ and $I = \langle f_1, f_2, \cdots, f_\ell \rangle$ be an ideal such that each f_i is in $\mathbb{F}[x_1, x_2, \cdots, x_k]$. Then the following easy lemma states a necessary and sufficient condition for f to be in I.

Lemma 3.4.3 *Let $I \subseteq \mathbb{F}[\bar{x}]$ be an ideal generated by the polynomials $f_1, f_2, \cdots, f_\ell$ such that for all $i \in [\ell]$, $f_i \in \mathbb{F}[x_1, x_2, \cdots, x_k]$. Let g be any polynomial in $\mathbb{F}[\bar{x}]$. Write $g = \sum_{\bar{\alpha} \in \mathbb{N}^{n-k}} A_{\bar{\alpha}} \bar{x}^{\bar{\alpha}}$. Then $g \in I$ if and only if for all $\bar{\alpha}$, $A_{\bar{\alpha}} \in I$.*

Consider polynomials $f, g \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ and an ideal I such that $g \in \langle I, f\rangle$. The following useful lemma gives a sufficient condition on f under which the remainder r obtained when we invoke Divide$(g; f)$ (of Theorem 3.2.2) is in the ideal I.

Lemma 3.4.4 *Let $I = \langle f_1, f_2, \cdots, f_\ell\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ where the generators $f_i \in \mathbb{F}[x_1, \cdots, x_k]$. Let R denotes the polynomial ring $\mathbb{F}[x_1, \cdots, x_k]$. Suppose f is a polynomial such that $LM(f)$ contains only variables from $\{x_{k+1}, x_{k+2}, \cdots, x_n\}$ (i.e. $LM(f) = LM_R(f)$). Then for any polynomial g in the ideal $\langle I, f\rangle$ we can write $g = qf + r$ for polynomials q and r such that $r \in I$ and no monomial of r is divisible by $LM(f)$.*

Proof. The lemma is an easy consequence of the properties of the Divide algorithm explained in Theorem 3.2.2. Notice that Divide$(g; f)$ will stop with a remainder polynomial r such that $g = qf + r$ with the property that no monomial of r is divisible by $LM(f)$. However, we only know that $r \in \langle I, f\rangle$, because both g and qf are in $\langle I, f\rangle$. We now show that r must be in I. First, as $r \in \langle I, f\rangle$ we can write $r = \sum_{i=1}^{\ell} a_i f_i + af$, for polynomials a_i and a. Following Lemma 3.4.3, we write $a_i = \sum_{\bar{\alpha}} a_{i\bar{\alpha}} \bar{x}^{\bar{\alpha}}$ for each i and also $a = \sum_{\bar{\alpha}} a_{\bar{\alpha}} \bar{x}^{\bar{\alpha}}$. Notice that we can assume $a_{\bar{\alpha}} \notin I$ for all nonzero $a_{\bar{\alpha}}$. Otherwise, we can move that term to the $\sum a_i f_i$ part. Since $LM(f)$ does not divide any monomial of r, it follows that $LM(af)$ does not occur in a nonzero term of r. Therefore, $LT(af)$ must be cancelled by some term of $\sum_{i=1}^{\ell} a_i f_i$. Clearly, $LT(af)$ is of the form $c \cdot a_{\bar{\beta}} \bar{x}^{\bar{\alpha}}$ for some α, β, where $LC(f) = c \in \mathbb{F}$ and $a_{\bar{\beta}} = LC_R(a)$. Now, in $\sum_{i=1}^{\ell} a_i f_i$ the coefficient of $\bar{x}^{\bar{\alpha}}$ is $\sum_{i=1}^{\ell} a_{i\bar{\alpha}} f_i$ which must be equal to $-c \cdot a_{\bar{\beta}}$. Since $c \in \mathbb{F}$ it follows that $a_{\bar{\beta}}$ is in I contradicting the assumption that none of the nonzero $a_{\bar{\gamma}}$ is in I. ■

Again, let $I = \langle f_1, f_2, \cdots, f_\ell\rangle$ such that the f_i are in $\mathbb{F}[x_1, x_2, \cdots, x_k]$. Consider two polynomials f and g such that $LM(f)$ contains only variables from $x_{k+1}, x_{k+2}, \cdots, x_n$ and either $LM(f) > LM(g)$ or $LM_R(f) = LM_R(g)$ and $LC_R(g) \in I$. Then g is in the ideal $\langle I, f\rangle$ if and only if $g \in I$.

Lemma 3.4.5 *Let $I = \langle f_1, f_2, \cdots, f_\ell\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ such that each f_i is in $\mathbb{F}[x_1, x_2, \cdots, x_k] = R$. Suppose f is a polynomial such that $LM(f)$ is over the variables only from $\{x_{k+1}, x_{k+2}, \cdots, x_n\}$ (i.e. $LM(f) = LM_R(f)$). Then for any polynomial g such that either $LM(f) > LM(g)$, or $LM_R(f) = LM_R(g)$ and $LC_R(g) \in I$, g is in the ideal $\langle I, f\rangle$ if and only if g is in the ideal I.*

Proof. Suppose $g \in \langle I, f\rangle$ and $g \notin I$. We can write $g = a + bf$, for polynomials a and b, where $a \in I$. Also, we can assume that $b \notin I$, for otherwise $g \in I$ and we are done. Let $b = \sum_{\bar{\alpha}\in\mathbb{N}^{n-k}} b_{\bar{\alpha}}\bar{x}^{\bar{\alpha}}$, where $b_{\bar{\alpha}} \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ and we can assume $b_{\bar{\alpha}} \notin I$ for all $\bar{\alpha}$ (otherwise we can move that term as part of a). Notice that $LT_R(bf) = LT_R(b) \cdot LT_R(f) = cb_{\bar{\beta}}LM_R(b)LM_R(f) = cb_{\bar{\beta}}\bar{x}^{\bar{\gamma}}$ for some $\bar{\gamma}$ and for some $b_{\bar{\beta}}$, where $c = LC_R(f) \in \mathbb{F}$. Since $b_{\bar{\beta}} \notin I$ it follows that $LC_R(bf) \notin I$. Write $a = \sum_{\bar{\alpha}\in\mathbb{N}^{n-k}} a_{\bar{\alpha}}\bar{x}^{\bar{\alpha}}$. By Lemma 3.4.3, $a \in I$ implies each $a_{\bar{\alpha}} \in I$. In particular, $a_{\bar{\gamma}} \in I$ and is *not* equal to $-LC_R(b \cdot f) = -cb_{\bar{\beta}}$ as $b_{\bar{\beta}} \notin I$. Thus, the monomial $LM_R(bf)$ survives in $a + bf$. It follows that $LM_R(g) = LM_R(a + bf) \geq LM_R(bf) \geq LM_R(f)$ which forces $LM_R(f) = LM_R(g)$ and $LC_R(g) \in I$ by assumption. If $b \notin R$ then $LM_R(b \cdot f) > LM_R(f)$ which implies $LM_R(g) > LM_R(f)$ contradicting assumption. If $b \in R$ then $LT_R(g) = LT_R(a + bf) = (a_{\bar{\alpha}} + b)LM_R(f)$ for some $a_{\bar{\alpha}}$, which forces $b \in I$ because both $LT_R(g)$ and $a_{\bar{\alpha}} \in I$. ■

Let $I \subseteq \mathbb{F}[x_1, \cdots, x_n]$ be an ideal and g_1, g_2 are two polynomials such that f is in the ideals $\langle I, g_1\rangle$ and $\langle I, g_2\rangle$. Using some Gröbner basis theory we give a sufficient condition on I, g_1 and g_2 under which we can infer that f is in the ideal $\langle I, g_1g_2\rangle$.

Lemma 3.4.6 *Let $I = \langle f_1, f_2, \cdots, f_\ell\rangle$ be an ideal of $\mathbb{F}[x_1, x_2, \cdots, x_n]$, where f_i are polynomials in $\mathbb{F}[x_1, x_2, \cdots, x_k]$. Suppose g_1 and g_2 are polynomials such that: $g_2 = \prod_{i=1}^{d_2}(x_{k+1} - \alpha_i)$, where each α_i is a linear form over $x_1, x_2, \cdots, x_k$, and the leading term $LT(g_1)$ of g_1 has only variables from $\{x_{k+2}, x_{k+3}, \cdots, x_n\}$. Then $f \in \langle I, g_1g_2\rangle$ if and only if $f \in \langle I, g_1\rangle$ and $f \in \langle I, g_2\rangle$.*

Proof. The forward implication is obvious. We prove the reverse direction. Suppose $f \in \langle I, g_1\rangle$ and $f \in \langle I, g_2\rangle$. As $f \in \langle I, g_2\rangle$, we can write $f = a + bg_2$, where $a \in I$ and b is an arbitrary polynomial. Notice that it suffices to prove bg_2 is in the ideal $\langle I, g_1g_2\rangle$. Now, since $f \in \langle I, g_1\rangle$ and $a \in I$ it follows that $bg_2 = f - a \in \langle I, g_1\rangle$. By applying Lemma 3.4.4 to ideal I and polynomial g_1 observe that we can write $bg_2 = \alpha g_1 + \beta$, where β is a polynomial in I such that none of the monomials of β is divisible by $LT(g_1)$. We have the following equation $b \cdot \prod_{j=1}^{d_2}(x_{k+1} - \alpha_j) = \alpha g_1 + \beta$.

Substituting $x_{k+1} = \alpha_1$ in the above equation, we get $(\alpha g_1)|_{x_{k+1}=\alpha_1} = -\beta|_{x_{k+1}=\alpha_1}$. Notice that $LT(g_1|_{x_{k+1}=\alpha_1}) = LT(g_1)$. This is because, $LT(g_1)$ contains variables only from $x_{k+2}, \cdots, x_n$. Thus the above substitution implies,

$$LT(\beta|_{x_{k+1}=\alpha_1}) = -LT((\alpha g_1)|_{x_{k+1}=\alpha_1}) = -LT(\alpha|_{x_{k+1}=\alpha_1}) \cdot LT(g_1|_{x_{k+1}=\alpha_1})$$

and,

$$LT(g_1|_{x_{k+1}=\alpha_1}) = \ - LT(\alpha|_{x_{k+1}=\alpha_1}) \cdot LT(g_1).$$

Thus $LM(g_1)$ divides $LM(\beta|_{x_{k+1}=\alpha_1})$. On the other hand, since $LM(g_1)$ does not divide any monomial of β, $LM(g_1)$ cannot divide any monomial of $LM(\beta|_{x_{k+1}=\alpha_1})$ as the substitution only introduces variables from $\{x_1, \cdots, x_k\}$. This gives a contradiction unless $\beta|_{x_{k+1}=\alpha_1} = 0$, which in turn implies $\alpha|_{x_{k+1}=\alpha_1} = 0$.

Thus we have proved that $(x_{k+1} - \alpha_1)$ is a factor of both α and β. This leads us to the following similar identity: $b \cdot \prod_{j=2}^{d_2}(x_{k+1} - \alpha_j) = \alpha_1 g_1 + \beta_1$, where $\alpha_1 = \alpha/(x_{k+1} - \alpha_1)$ and $\beta_1 = \beta/(x_{k+1} - \alpha_1)$. Clearly, by repeating the above argument we finally get, $b = \alpha' g_1 + \beta'$, for some polynomials α' and β' where $\alpha = \alpha' g_2$ and $\beta = \beta' g_2$. Putting it together we get $bg_2 = \alpha' g_1 g_2 + \beta' g_2 = \alpha' g_1 g_2 + \beta$. As $\beta \in I$, it follows that bg_2 is in the ideal $\langle I, g_1 g_2\rangle$. This completes the proof. ■

Let $I = \langle P_1, P_2, \cdots, P_k\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ such that $P_i \in \mathbb{F}[x_1, x_2, \cdots, x_i]$ and $LT(P_i) = x_i^{d_i}$ for each i. For $i \neq j$ the leading terms $LT(P_i) = x_i^{d_i}$ and $LT(P_j) = x_j^{d_j}$ are clearly relatively prime. Therefore by Theorem 3.2.6, it follows that $\{P_1, P_2, \cdots, P_k\}$ is in fact a Gröbner basis for I. We summarize this observation.

Lemma 3.4.7 *Let $I = \langle P_1, P_2, \cdots, P_k\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ such that each P_i is in $\mathbb{F}[x_1, x_2, \cdots, x_i]$ and $LT(P_i) = x_i^{d_i}$. Then $\{P_i\}_{i\in[k]}$ is a Gröbner basis for I.*

Let $f \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ be a given polynomial and d be the maximum of $\deg(f)$ and $\deg(P_i), 1 \leq i \leq k$. We can invoke Divide$(f; P_1, P_2 \cdots, P_k)$ (Theorem 3.2.2) to test whether $f \in I$. By Fact 3.2.3 the running time for this test is $O(d^k)$.

Now we state the main theorem of this section.

Theorem 3.4.8 *Let $C \in \mathbb{F}[x_1, x_2 \cdots, x_n]$ be given by a $\Sigma\Pi\Sigma(\ell, d)$ circuit for a constant ℓ and $I = \langle m_1, m_2, \cdots, m_k\rangle$ be a monomial ideal for constant k. For such instances, Monomial Ideal Membership can be checked in deterministic polynomial time. Specifically, the running time is bounded by n^kpoly$(n, d^{max\{\ell,k\}})$.*

By Lemma 3.3.1 it clearly suffices to give a polynomial-time deterministic algorithm for testing if a $\Sigma\Pi\Sigma(\ell, d)$ circuit C is in a monomial ideal of the form $\langle x_1^{e_1}, \cdots, x_k^{e_k}\rangle$. As explained in the beginning of this section, we transform the circuit C to C' in which all linear forms are made homogeneous using a new indeterminate y, and $C \in I$ if and only if $C' \in \langle x_1^{e_1}, \cdots, x_k^{e_k}, y-1\rangle$. In fact, in the following theorem we prove a stronger result which along with Lemma 3.3.1 yields Theorem 3.4.8.

Theorem 3.4.9 *Let C be a given $\Sigma\Pi\Sigma(\ell, d)$ circuit for a constant ℓ and $I = \langle P_1, P_2, \cdots, P_k\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ such that $P_i \in \mathbb{F}[x_1, x_2, \cdots, x_i]$ and $LT(P_i) = x_i^{d_i}$ for each i. Further, suppose $d_i \leq d$ for all $i \in [k]$. Then testing if $C \in I$ can be done deterministically in time* poly$(d^{max\{\ell,k\}})$.

Proof. We first describe the algorithm and then prove its correctness and running time bound.

As explained in the beginning of the section, we can assume that all linear forms appearing in C are homogeneous and C itself is a homogeneous degree d polynomial. By Lemma 3.4.7, the generating set for I is a Gröbner basis. Let $C(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{\ell} T_i$. For all $i \in [\ell]$, $T_i = \prod_{j=1}^{d} L_{ij}$, where L_{ij}'s are the linear forms over the ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$.

If $\ell = 1$, then $C = T_1$. Let $g(x_1, x_2, \cdots, x_k)$ be the product of those linear forms of T_1 using only variables from $\{x_1, x_2, \cdots, x_k\}$. Clearly, $g(x_1, x_2, \cdots, x_k)$ has at most d^k monomials. We explicitly compute g by multiplying out all such linear forms. By Lemma 3.4.3, clearly $C \in I$ if and only if $g \in I$, which can be checked in time poly(d^k) following the Fact 3.2.3.

So assume $\ell > 1$. If all the linear forms appearing in $T_1, T_2, \cdots, T_\ell$ are only over $\{x_1, x_2, \cdots, x_k\}$, then again the ideal membership testing is easy. Because, in time poly(d^k) we can write C itself as an $\mathbb{F}$-linear combination of monomials in $x_1, x_2, \cdots, x_k$ and apply Fact 3.2.3 to check if $f \in I$ in time poly(d^k).

Now we consider the general case. By inspection we can write each $T_i = \beta_i T_i'$ where the β_i are products of linear forms over only $x_1, x_2, \cdots, x_k$, whereas each linear form in T_i' involves at least one other variable.[2] If $\beta_i \in I$ (which we can test in polynomial time using Fact 3.2.3) we drop the term T_i from the sum $\sum_{i=1}^{\ell} T_i$. This enables us to write C as $C = \beta_1 T_1' + \beta_2 T_2' + \cdots + \beta_m T_m'$ for some $m \leq \ell$, where we have assumed for simplicity of notation that $\beta_i \notin I$ for first m terms.

As before, let $R = \mathbb{F}[x_1, x_2, \cdots, x_k]$. W.l.o.g, assume that $LM_R(T_1') \geq LM_R(T_i')$ for all $i \in [2, 3, \cdots, m]$. We can determine $LT_R(T_i')$ for each T_i' in polynomial time since they are given as product of linear forms. Thus, $LM_R(T_1') \geq LM_R(C)$. Now,

[2]If there are no linear forms contributing to the product β_i (respectively, T_i') we will set it to 1.

let $r \in R$ be the coefficient of $LM_R(T_1')$ in C. We can compute r in polynomial time by computing the coefficient γ_i of $LM_R(T_1')$ in each T_i' and computing $r = \sum_{i=1}^{m} \beta_i \gamma_i$. Then we check that $r \in I$ (which is a necessary condition for C to be in I by Lemma 3.4.3). By Fact 3.2.3 we can check $r \in I$ in time poly(d^k). It is clear that, either $LM_R(T_1') > LM_R(C)$ or $LM_R(T_1') = LM_R(C)$ and $r \in I$. Thus, by Lemma 3.4.5, $C \in I$ if and only if $C \in \langle I, T_1' \rangle$.

Next, we group the linear forms in T_1': let, $T_1' = T_{11}T_{12}\cdots T_{1t}$, such that for all $i \in [t]$,

$$T_{1i} = (L_i + m_{i1})(L_i + m_{i2}) \cdots (L_i + m_{is_i}),$$

where $\{L_i\}_{i=1}^{t}$ are *distinct linear forms* in $\mathbb{F}[x_{k+1}, \cdots, x_n]$ and m_{ij}'s are linear forms in $\mathbb{F}[x_1, \cdots, x_k]$. Notice that the polynomials T_{1i} are relatively prime to each other.

We next compute t linear transformations $\{\sigma_1, \sigma_2, \cdots, \sigma_t\}$ from $\mathbb{F}^n$ to $\mathbb{F}^n$ with the following property: for $i \in [t]$, σ_i fixes $\{x_i\}_{i=1}^{k}$, maps L_i to x_{k+1} and maps the variables $x_{k+2}, x_{k+3}, \cdots, x_n$ to some suitable linear forms in such a way that, σ_i is an invertible linear transformation. As L_i's are over $\{x_{k+1}, \cdots, x_n\}$, it is easy to see that such σ_i exist and are easy to compute.

Let $C_1 = \sum_{j \in [\ell] \setminus \{1\}} T_j$. For $i \in [t]$, let $C_{1i} = \sigma_i(C_1)$ and let I_{1i} be the ideal $\langle I, \sigma_i(T_{1i}) \rangle$. The algorithm will now recursively check for each of the $\Sigma\Pi\Sigma(\ell - 1, d)$ circuits C_{1i}, that C_{1i} is in the ideal I_{1i} and declare $C \in I$ if and only if $C_{1i} \in I_{1i}$ for each i.

Notice that the ideal I_{1i} has generating set $G = \{P_1, P_2, \cdots, P_k, P_{k+1}\}$, where $P_{k+1} \in \mathbb{F}[x_1, x_2, \cdots, x_{k+1}]$ and $LM(P_{k+1}) = x_{k+1}^{d_{k+1}}$. By Lemma 3.4.7, G is a Gröbner basis for I_{1i}.

The correctness of the algorithm follows directly from the following claim.

Claim 3.4.10 *For each $s : 1 \leq s \leq t$ $C \in \langle I, T_{11}T_{12}\cdots T_{1s} \rangle$ if and only if $C_{1i} \in I_{1i}$ for $1 \leq i \leq s$.*

In particular, $C \in \langle I, T_1' \rangle$ if and only if $C_{1i} \in I_{1i}$ for $1 \leq i \leq t$.

Proof of Claim: The forward implication is easy: if $C \in \langle I, T_{11}T_{12}\cdots T_{1s} \rangle$ then clearly $C \in \langle I, T_{1i} \rangle$ for each $1 \leq i \leq s$. As each σ_i is an invertible linear map it follows in turn that $\sigma_i(C) \in \langle I, \sigma_i(T_{1i}) \rangle = I_{1i}$ for $1 \leq i \leq s$. Since $C_{1i} = \sigma_i(C) - \sigma_i(T_1)$ and $\sigma_i(T_1) \in \langle \sigma_i(T_{1i}) \rangle$ it follows that $C_{1i} \in I_{1i}$ for $1 \leq i \leq s$.

We prove the other direction of the claim by induction on s. The base case $s = 1$ is trivial. Inductively assume it is true for $s-1$. I.e. if $C_{1i} \in I_{1i}$ for $1 \le i \le s-1$ then $C \in \langle I, T_{11}T_{12}\cdots T_{1(s-1)}\rangle$.

We now prove the induction step for s. Suppose $C_{1i} \in I_{1i}$ for $1 \le i \le s$. Let $T = T_{11}T_{12}\cdots T_{1(s-1)}$. By induction hypothesis we have $C \in \langle I, T\rangle$. Furthermore, $C_{1s} \in I_{1s}$ implies by definition that $C \in \langle I, T_{1s}\rangle$. Now we apply the linear map σ_s to obtain $\sigma_s(C) \in \langle I, \sigma_s(T)\rangle$ and $\sigma_s(C) \in \langle I, \sigma_s(T_{1s})\rangle$. The map σ_s ensures that $LT(T_{1s})$ is of the form $x_{k+1}^{\deg T_{1s}}$. Furthermore, by the definition of σ_s it follows that $LT(\sigma_s(T))$ has only variables in $\{x_{k+2}, \cdots, x_n\}$. Letting $g_1 = \sigma_s(T)$ and $g_2 = \sigma_s(T_{1s})$ in Lemma 3.4.6, it follows immediately that $\sigma_s(C) \in \langle I, \sigma_s(T \cdot T_{1s})\rangle$ which implies the induction step since σ_s is invertible.

Claim 3.4.11 *The above algorithm runs in time* poly$(n, d^{\max\{\ell,k\}})$.

Proof of Claim: To analyze the running time, we need to observe the following recurrence relation : let $T(\ell, n)$ is the time required to test $C \in I$. It is easy to see from the description of the algorithm that, $T(\ell, n) \le tT(\ell-1, n) +$ poly(n, d^k). Hence $T(\ell, n) =$ poly$(n, d^{\max\{\ell,k\}})$, as $t = O(d)$. ■

Theorem 3.4.8 is an immediate consequence of Theorem 3.4.9. For $I = \langle 0\rangle$, Theorem 3.4.8 is actually the Kayal-Saxena deterministic test with a new proof.

3.5 Monomial Ideal Membership for black-box polynomials

In Theorem 3.3.3 we have shown that monomial ideal membership is in randomized polynomial time when $f \in \mathbb{F}[\bar{x}]$ is given as an arithmetic circuit and the monomial ideal is given by a constant number of generator monomials. We now show that even if f is accessed only via a *black-box*, if the degree of f is *polynomial in the input size* we can still solve monomial ideal membership in randomized polynomial time (assuming I is generated by constant number of monomials). In [BOT88], Ben-Or and Tiwari gave an interpolation algorithm for sparse multivariate polynomials over integers. Our algorithm is an easy application of their result. We first recall their result in a form suitable for us.

Theorem 3.5.1 [BOT88] *Let $f \in \mathbb{Z}[x_1, x_2, \cdots, x_n]$ be a t-sparse multivariate polynomial given as a black-box (by t-sparse we mean the number of monomials in f is* *bounded by t), d be the degree of f, and b be a bound on the size of its coefficients. There is a deterministic algorithm that queries the black-box for values of f on different inputs and reconstructs the entire polynomial f in time* $\mathrm{poly}(t, n, d, b)$.

Ben-Or and Tiwari's result directly gives a deterministic polynomial time algorithm for Monomial Ideal Membership when f is a t-sparse black-box polynomial over $\mathbb{Z}$, and I is any monomial ideal. The algorithm simply reconstructs f and checks if each of its monomials is in I.

Next, suppose f is a black-box polynomial of small degree and I is a monomial ideal generated by constant number of monomials.

Theorem 3.5.2 *Let $f \in \mathbb{Z}[\bar{x}]$ of degree d given as a black-box such that b is a bound on the size of its coefficients. Suppose $I = \langle m_1, m_2, \cdots, m_k\rangle$ for constant k. Then we can test if $f \in I$ in randomized time* $\mathrm{poly}(n^k, d^k, b)$.

Proof. By Lemma 3.3.1, it suffices to give a randomized polynomial time algorithm for testing if $f \in I_{\bar{v}}$, where $\bar{v} \in [n]^k$. W.l.o.g. assume $I_{\bar{v}} = \langle x_1^{e1}, x_2^{e2}, \cdots, x_k^{e_k}\rangle$. Fix $S = \{1, 2, \cdots, s\}$ and assign random values $\bar{r} = \{r_1, r_2, \cdots, r_{n-k}\}$ to $\{x_{k+1}, \cdots, x_n\}$ from S. Note that $f(x_1, x_2, \cdots, x_k, \bar{r})$ is a d^k-sparse polynomial. By Theorem 3.5.1 we can reconstruct $f(x_1, x_2, \cdots, x_k, \bar{r})$ in poly(n, d^k, b) time. Let $g(x_1, x_2, \cdots, x_k) = f(x_1, x_2, \cdots, x_k, \bar{r})$. Our randomized algorithm declares $f \in I_{\bar{v}}$ if each monomial of g is in I. By Lemma 3.3.2, it follows that the success probability of the algorithm is at least $1 - \frac{d}{|S|}$. So it is enough to consider $S = \{1, 2, \cdots, 2d\}$ and the number of random bits used is $O(n \log d)$. ■

3.6 Bounded variable Ideal Membership

In this section we discuss our results for the ideal membership problem when $I = \langle f_1, \cdots, f_m\rangle$ such that $f_i \in \mathbb{F}[x_1, \cdots, x_k]$ for a constant k and the polynomial f is given by an arithmetic circuit. We call this variant *bounded variable Ideal Membership*.

A pioneering result in polynomial Ideal Membership testing is Hermann's algorithm that is based on the following theorem.

Theorem 3.6.1 [Her26, Sud98, Hermann's Theorem] *Consider the multivariate polynomials $f, f_1, f_2, \cdots, f_m \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ such that,*

$$\max\{\deg(f_1), \deg(f_2), \cdots, \deg(f_m), \deg(f)\} \leq d.$$

If f is in the ideal $I = \langle f_1, f_2, \cdots, f_m\rangle$ then f can be expressed as $f = \sum_{i=1}^{m} g_i f_i$ where $\deg(g_i) \leq (2d)^{2^k}$ for each i.

Suppose f is given explicitly as an $\mathbb{F}$-linear combination of terms. Using the bounds of Hermann's theorem, Hermann's algorithm treats the coefficients of g_i as unknowns and does membership testing in $\langle f_1, f_2, \cdots, f_m\rangle$ by solving a system of linear equations with $m(2d)^{k2^k}$ unknowns. This can be solved using Gaussian elimination in time $m^{O(1)}(2d)^{O(k2^k)}$.

Similarly, for an explicitly given $f \in \mathbb{F}[x_1, \cdots, x_n]$, $n > k$, using Lemma 3.4.3 we can apply Hermann's algorithm to test if membership of f in $\langle f_1, f_2, \cdots, f_m\rangle$ in time polynomial in the size of f and $m^{O(1)}(2d)^{O(k2^k)}$. If k is a constant, this gives a polynomial running time bound.

A natural question here is the complexity of Ideal Membership when f is given by an arithmetic circuit whose membership we want to test in ideal $I = \langle f_1, f_2, \cdots, f_m\rangle$, where $f_i \in \mathbb{F}[x_1, \cdots, x_k]$ for constant k. Recall that in Theorem 3.3.3 we showed a similar problem for *monomial* ideals with constant number of monomials is in randomized polynomial time. In this section we will restrict ourselves to polynomials f computed by arithmetic circuits of polynomial-degree in the input size. We can follow essentially the same proof idea in Theorem 3.3.3. Notice that $f \in I$ if and only if $f \equiv 0$ in the ring $R[x_{k+1}, x_{k+2}, \cdots, x_n]$ where $R = \mathbb{F}[x_1, x_2, \cdots, x_k]/I$. We recall the following results (Proposition 2.6.1 and Lemma 2.6.3) from Chapter 2.

Proposition 3.6.2 *Let R be a finite commutative ring with unity 1 containing a field $\mathbb{F}$ such that $1 \in \mathbb{F}$. If $f \in R[x]$ is a nonzero polynomial of degree d then $f(a) = 0$ for at most d distinct values of $a \in \mathbb{F}$.*

Lemma 3.6.3 *Let R be a finite commutative ring with unity 1 containing a field $\mathbb{F}$ such that $1 \in \mathbb{F}$. Let $g \in R[x_1, x_2, \cdots, x_m]$ be any polynomial of degree at most d. If $g \not\equiv 0$, then for any finite subset A of $\mathbb{F}$ we have*

$$\text{Prob}_{a_1 \in A, \cdots, a_m \in A}[g(a_1, a_2, \cdots, a_m) = 0 \mid g \not\equiv 0] \leq \frac{d}{|A|}.$$

Now we describe our ideal membership test: Let $\mathbb{F}$ be a finite field. Choose and fix $S \subseteq \mathbb{F}$ of size $2d$ and randomly assign values from S to the variables in $\{x_{k+1}, \cdots, x_n\}$. Notice that f, given by a polynomial degree arithmetic circuit C, is in I if and only if $f \equiv 0$ in the ring $R[x_{k+1}, x_{k+2}, \cdots, x_n]$ where $R = \mathbb{F}[x_1, x_2, \cdots, x_k]/I$, since the given generating set for I uses only variables $x_1, \cdots, x_k$. After the random substitution we are left with an arithmetic circuit $C'(x_1, \cdots, x_k)$. Notice that, by Lemma 3.6.3 if $f \notin I$ then $C'(x_1, \cdots, x_k) \notin I$ with probability at least $1/2$. We now need to test whether the polynomial computed by C' is in I. As C' is of polynomial degree d and k is a constant, we can explicitly written down the polynomial r that it computes as a $\mathbb{F}$-linear combination of at most d^k monomials. We are now left with the problem of testing if $r \in \langle f_1, \cdots, f_m \rangle$ which we can do in polynomial time using Hermann's algorithm as k is a constant. It is easy to that, same arguement works when $\mathbb{F}$ is an infinite field, in particular $\mathbb{F} = \mathbb{Q}$.

Now, we consider the case when the input polynomial $f \in \mathbb{Z}[x_1, x_2, \cdots, x_n]$ is given by a black-box and the degree of f is at most d. Let $I = \langle f_1, f_2, \cdots, f_m \rangle$ be the given ideal where $f_i \in \mathbb{Z}[x_1, x_2, \cdots, x_k]$. So, $f \in I$ if and only if $f \equiv 0$ in $R[x_{k+1}, x_{k+2}, \cdots, x_n]$ where $R = \mathbb{Z}[x_1, x_2, \cdots, x_k]/I$. Choose and fix $S \subset \mathbb{Z}$ of size $2d$ and randomly assign values to $x_{k+1}, x_{k+2}, \cdots, x_n$ from S. Let $g(x_1, x_2, \cdots, x_k) = f(x_1, x_2, \cdots, x_k, \bar{a})$, where $\bar{a} \in S^{n-k}$ be a substitution for $(x_{k+1}, \cdots, x_n)$. Again, by Lemma 2.6.3, if $f \notin I$ then $g(x_1, x_2, \cdots, x_k) \not\equiv 0$ in R with probability at least $1/2$. Notice that g is d^k-sparse polynomial (k is a constant). Also, if b be the size of the coefficients of f, the size of the coefficients of g are bounded by poly(b, d). So, using Theorem 3.5.1, we can interpolate the polynomial g as a sum of monomials in polynomial time. Now, as before, using Hermann's algorithm we can test whether $g \in I$ in polynomial time.

Finally, we consider the case when f is given by a $\Sigma\Pi\Sigma(\ell, d)$ circuit C with fan-in of output gate (ℓ) a constant. We can easily argue by following the algorithm in the proof of Theorem 3.4.9 that we can test in deterministic polynomial time whether $f \in I$. We briefly describe the algorithm. As before, $I = \langle f_1, f_2, \cdots, f_m \rangle$, where $f_i \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ and k is a constant. Let $C = \sum_{i=1}^{\ell} T_i = \sum_{i=1}^{\ell} \prod_{j=1}^{d} L_{ij}$. If $\ell = 1$, then $C = T_1$. Let $\hat{g}(x_1, x_2, \cdots, x_k)$ be the products of those linear forms using only variables from $x_1, x_2, \cdots, x_k$. By Lemma 3.4.3, $C \in I$ if and only if $\hat{g} \in I$. We write $\hat{g}$ explicitly as a sum of at most d^k monomials. Now, using Hermann's algorithm we can test whether $\hat{g} \in I$ in deterministic polynomial time as k is a constant. [3]

Again, the case that $\ell > 1$ and $T_i \in \mathbb{F}[x_1, x_2, \cdots, x_k]$, can be handled easily using Hermann's algorithm.

Now we consider the general case when $T_i = \beta_i T_i'$, where β_i is the product of the linear forms over $x_1, x_2, \cdots, x_k$ and each linear form in T_i' involves at least one other variable. We follow the algorithm described in the proof of Lemma 3.4.9 closely. It is easy to see that whenever we reduce the problem size by removing a product gate T_i, we grow the corresponding ideal by putting a new polynomial in its generating set.

Following the proof of Lemma 3.4.9 step by step, it is clear that we will end up with the problem of testing if a polynomial g given by a $\Pi\Sigma$ circuit is in an ideal $\langle g_1, \cdots, g_w\rangle$, where g_i are all in $\mathbb{F}[x_1, \cdots, x_t]$ for a constant t and w is a polynomial in the input size. Invoking Hermann's algorithm, we can check this in time poly$(w, d^{O(t2^t)})$, which is a polynomial time bound as t is constant. We summarize this result in the following theorem.

Theorem 3.6.4 *Let $I = \langle f_1, f_2, \cdots, f_m\rangle$ be an ideal in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ where each $f_i \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ for constant k. If f be a polynomial given by an arithmetic circuit of polynomial degree, then in randomized polynomial time we can test if $f \in I$. This result holds even if f is given by a black-box and the degree of f is polynomial in the input size. Further, if f is given by a $\Sigma\Pi\Sigma(\ell, d)$ circuit with ℓ constant, then we can test whether $f \in I$ in deterministic polynomial time.*

3.7 Identity Testing for a restricted class of $\Sigma\Pi\Sigma\Pi$ circuits

In this section, we examine the possibility of extending [KS07] to certain depth 4 circuits. We consider certain restricted $\Sigma\Pi\Sigma\Pi$ circuits with the top Σ gate having bounded fan-in.

Any $\Sigma\Pi\Sigma\Pi$ circuit is of the form $C = \sum_{i=1}^{\ell} T_i$, with $T_i = \prod_{j=1}^{d} P_{ij}$, for polynomials P_{ij}. We now define a restricted class of depth 4 circuits which we denote by $\Sigma\Pi\Sigma\Pi(\ell, d, c)$. A circuit C is in this class if

[3]The set of polynomials $f_1, f_2, \cdots, f_m$ is not a Gröbner basis for I in general. So we can not invoke Divide algorithm directly.

(a) The fan-in ℓ of the output Σ gate is a constant.

(b) For each variable x_k occurring in P_{ij}'s, the term of maximum x_k degree is a power of x_k only.

(c) Any variable x_k occurs in at most c different P_{ij} for any $i \in [\ell]$, where c is also a constant.

(d) Furthermore, each P_{ij} contains at most c different variables.

We show that the bounded variable Ideal Membership problem for $\Sigma\Pi\Sigma\Pi(\ell, d, c)$ circuits can be solved in polynomial time. As a consequence we obtain a deterministic polynomial-time identity testing algorithm for such circuits. The key observation is the next lemma which generalizes Lemma 3.4.6.

Lemma 3.7.1 *Let $I = \langle f_1, f_2, \cdots, f_\ell \rangle$ be an ideal of $\mathbb{F}[x_1, x_2, \cdots, x_n]$, where f_i are polynomials in $\mathbb{F}[x_1, \cdots, x_k]$. Suppose g_1 and g_2 are the polynomials such that:*

1. $LM(g_1) = x_i^{d_i}$, where $i \in \{k+1, k+2, \cdots, n\}$.

2. $LM(g_2) < LM(g_1)$ and $LM(g_2), LM(g_1)$ are relatively prime.

Then $f \in \langle I, g_1 \rangle$ and $f \in \langle I, g_2 \rangle$ if and only if $f \in \langle I, g_1 g_2 \rangle$.

Proof. The reverse implication is obvious. We prove the forward direction. As $LM(g_2) < LM(g_1)$ and $LM(g_2), LM(g_1)$ are relatively prime, it follows that $g_2 \in \mathbb{F}[x_1, x_2, \cdots, x_{i-1}]$.

As $f \in \langle I, g_2 \rangle$, we can write $f = a + bg_2$, where $a \in I$ and b is an arbitrary polynomial. Furthermore, by Lemma 3.4.4 we can write $bg_2 = \alpha g_1 + \beta$, with $\beta \in I$ such that no monomial of β is divisible by $LT(g_1)$. Thus g_2 divides $\alpha g_1 + \beta$. Let p be any irreducible factor of g_2. As the ideal $\langle p \rangle$ generated by the polynomial p is a prime ideal of $R = \mathbb{F}[x_1, x_2, \cdots, x_{i-1}]$, the quotient ring $D = R/\langle p \rangle$ is an integral domain. As p divides $\alpha g_1 + \beta$, it follows that $\alpha g_1 = -\beta$ in $D[x_i]$. We will now argue that β and α must be both zero in $D[x_i]$, which will imply that p divides both α and β. Note that $LM_D(\beta) = -LM_D(\alpha) \cdot LM_D(g_1)$ (by comparing their x_i degrees in the ring $D[x_i]$). But $LM_D(g_1) = LM(g_1) = x_i^{d_i}$ from the statement of the lemma. Considering β as a polynomial of $R[x_i]$, notice that β has degree strictly less than d_i since $LM(g_1) = x_i^{d_i}$ does not divide any monomial of β. Since $p \in R = \mathbb{F}[x_1, x_2, \cdots, x_{i-1}]$, it follows that

β as a polynomial of $D[x_i]$ also has degree strictly less than d_i. Thus, $LM_D(g_1)$ can not divide $LM_D(\beta)$. The only possibility left is that $\alpha = \beta = 0$ in $D[x_i]$, which implies that p divides α and β.

This leads us to the following similar identity: $bg_2' = \alpha_1 g_1 + \beta_1$, where $\alpha_1 = \alpha/p$ and $\beta_1 = \beta/p$. Clearly, by the same argument applied to each irreducible factor of g_2 (with repetition) we finally get $b = \alpha' g_1 + \beta'$, for polynomials α' and β' where $\alpha = \alpha' g_2$ and $\beta = \beta' g_2$. Putting it together, $bg_2 = \alpha' g_1 \cdot g_2 + \beta' g_2 = \alpha' g_1 \cdot g_2 + \beta$. As $\beta \in I$, it follows that bg_2 is in the ideal $\langle I, g_1 g_2\rangle$. This completes the proof. ∎

Now we present the polynomial time algorithm for bounded variable ideal membership instances (f, I), where the polynomial f is given by a $\Sigma\Pi\Sigma\Pi(\ell, d, c)$ circuit. The polynomial-time identity test for $\Sigma\Pi\Sigma\Pi(\ell, d, c)$ circuits is a corollary.

Theorem 3.7.2 *Let C be a given $\Sigma\Pi\Sigma\Pi(\ell, d, c)$ circuit and $I = \langle f_1, f_2, \cdots, f_m\rangle$ be an ideal in $\mathbb{F}[x_1, \cdots, x_n]$ such that each $f_i \in \mathbb{F}[x_1, x_2, \cdots, x_k]$ where k is a constant. Then testing if $C \in I$ can be done deterministically in time* poly(n, d).

Proof. We first write $C = T_1 + T_2 + \cdots + T_\ell$, where each $T_i = \prod_{j=1}^{d} P_{ij}$. The case $\ell = 1$ and the case when each T_i is only over indeterminates $x_1, \cdots, x_k$ can be directly handled using Hermann's algorithm (Theorem 3.6.1), in time poly(d^{2^k}).

We describe the general case. Let $R = \mathbb{F}[x_1, x_2, \cdots, x_k]$. We can write $C = \beta_1 T_1' + \beta_2 T_2' + \cdots + \beta_m T_m'$ for some $m \le \ell$, where $\beta_i \in R$ and $\beta_i \notin I$, and T_i' are nontrivial polynomials in $R[x_{k+1}, \cdots, x_n]$. We can easily determine $LT_R(T_i')$ for each T_i' from the polynomials P_{ij}, and rearrange the T_i' so that $LM_R(T_1') \ge LM_R(T_2') \ge \cdots \ge LM_R(T_m')$. [4] Thus, $LM_R(T_1') \ge LM_R(C)$. The coefficient r of $LM_R(T_1')$ in C is also easily computable in polynomial time: we find the coefficient γ_i of $LM_R(T_1')$ in T_i' for $i = 1, 2, \cdots, m$. Note that $r = \sum_{i=1}^{m} \beta_i \gamma_i$. If $r \ne 0$ then notice that $r \notin I$ implies $C \notin I$. We check if $r \in I$ using Hermann's algorithm (Theorem 3.6.1) in time poly(d^{2^k}). We need to continue the test if $r \in I$. That means either $LM_R(T_1') > LM_R(C)$ or $LM_R(T_1') = LM_R(C)$ and $r \in I$. By Lemma 3.4.5, $C \in I$ if and only if $\sum_{i=2}^{m} \beta_i T_i' \in \langle I, T_1'\rangle$.

Next, we group the factors P_{ij} occurring in T_1' according to the leading monomials. Let T_{1r} be the product of all factors P_{1j} of T_1' such that $LM(P_{1j})$ is a power of x_r, for $r = k+1, k+2, \cdots, x_n$. For an index r if there are no such factors P_{1j} then set $T_{1r} = 1$.

[4]Notice the condition (b) in the definition of $\Sigma\Pi\Sigma\Pi(\ell, d, c)$ circuit.

Thus we have $T_1' = \prod_{r=k+1}^{n} T_{1r}$, where some of the factors T_{1r} are 1 and can be ignored. Clearly, for all $T_{1r} \neq 1$ and $T_{1s} \neq 1$ we have $LM(T_{1r}) > LM(T_{1s})$ if $r > s$.

Let $C_1 = \sum_{i=2}^{m} \beta_i T_i'$. For each r such that $T_{1r} \neq 1$, let I_{1r} denote the ideal $\langle I, T_{1r}\rangle$. Notice that T_{1r} is a polynomial over at most c^2 different variables. The algorithm recursively checks if C_1 is in the ideal I_{1r} for each ideal I_{1r} and declares $C \in I$ if and only if $C_1 \in I_{1i}$ for each i. Notice that C_1 is a $\Sigma\Pi\Sigma\Pi(\ell-1, d, c)$ circuit and the generators of I_{1i}'s are now over $k + c^2$ indeterminates (at most) which is still a constant.

Claim 3.7.3 *$C_1 = \sum_{i=2}^{m} \beta_i T_i' \in \langle I, T_1'\rangle$ if and only if $C_1 \in I_{1r}$ for each r such that $T_{1r} \neq 1$.*

Proof of Claim: We first write T_1' as $T_1' = T_{1i_1} T_{1i_2} \cdots T_{1i_t}$, where all $T_{1i_j} \neq 1$. Letting $g_2 = T_{1i_1} T_{1i_2} \cdots T_{1i_{t-1}}$ and $g_1 = T_{1i_t}$ in Lemma 3.7.1, we get that $C_1 \in \langle I, T_1'\rangle = \langle I, g_2 g_1\rangle$ if and only if $C_1 \in I_{1i_t}$ and $C_1 \in \langle I, T_{1i_1} T_{1i_2} \cdots T_{1i_{t-1}}\rangle$. A similar repeated application of Lemma 3.7.1 yields $C_1 \in \langle I, T_1'\rangle$ if and only if $C_1 \in \langle I, T_{1i_j}\rangle$ for each $j = 1, \cdots, t$. This completes the correctness proof of the algorithm.

We now show that the time bound is poly$(n, d^{max\{\ell, 2^k\}})$. Let $T(\ell, d, n)$ denote the time taken to test if $C \in I$. The algorithm description implies the following recurrence relation for T from which the running time bound is immediate.

$$T(\ell, d, n) \leq \begin{cases} dT(\ell, d, n) + \text{poly}(n, d^{2^k}) & \text{if } \ell > 1; \\ \text{poly}(n, d^{2^k}) & \text{if } \ell = 1. \end{cases}$$

■

4 Noncommutative Polynomial Identity Testing

4.1 Introduction

In this chapter we study polynomial identity problem over noncommutative model. To begin with, we state the identity testing problem over noncommutative model. Let C be a given arithmetic circuit computing a polynomial f in the noncommutative ring $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ where $\mathbb{F}$ is a field and x_i, x_j do not commute for $i \neq j$ (i.e. $x_i x_j - x_j x_i \neq 0$). Is there an efficient algorithm to test whether $f \equiv 0$ in $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$?

As shown by Nisan [Nis91] in noncommutative algebraic computation, proving lower bounds is somewhat easier. Using a rank argument Nisan has shown exponential size lower bounds for noncommutative formulas (and noncommutative algebraic branching programs) that compute the noncommutative permanent or determinant polynomials in the ring $\mathbb{F}\{x_1, \cdots, x_n\}$ where x_i are noncommuting variables. Thus, it seems plausible that identity testing in the noncommutative setting ought to be easier too. Indeed, Raz and Shpilka in [RS05] have shown that for noncommutative formulas (and algebraic branching programs) there is a deterministic polynomial-time algorithm for polynomial identity testing. However, for noncommutative circuits the situation is somewhat different. Bogdanov and Wee in [BW05] show using Amitsur-Levitzki's theorem that identity testing for *polynomial degree* noncommutative circuits is in randomized polynomial time. Basically, the Amitsur-Levitzki theorem allows them to randomly assign elements from a matrix algebra $M_k(\mathbb{F})$ for the noncommuting variables x_i, where $2k$ exceeds the degree of the circuit.

Our main contribution is the use of ideas from *automata theory* to design new efficient (deterministic) polynomial identity tests for *noncommutative* polynomials. More precisely, given a noncommutative circuit $C(x_1, \cdots, x_n)$ computing a polynomial of degree d with t monomials in $\mathbb{F}\{x_1, \cdots, x_n\}$, where the variables x_i are noncommuting, we give a deterministic polynomial identity test that checks if $C \equiv 0$ and runs in time polynomial in $d, |C|, n$, and t. The main idea in our algorithm is to think of the noncommuting monomials over the x_i as words and to design finite automata that allow us to distinguish between different words. Then, using the connection between automata, monoids and matrix rings we are able to deterministically choose a relatively small number of matrix assignments for the noncommuting variables to decide if $C \equiv 0$. Thus, we are able to avoid using the Amitsur-Levitzki theorem. Indeed, using our automata theory method we can easily give an alternative proof of (a weaker) version of Amitsur-Levitzki which is good enough for algorithmic purposes as in [BW05] for example.

Our method actually works in a black-box setting. In fact, given a noncommuting black-box polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ of degree d with t monomials, which we can evaluate by assigning matrices to x_i, we can reconstruct the entire polynomial f in time polynomial in n, d and t.

Furthermore, we also apply this idea to *black-box* noncommuting algebraic branching programs. We extend the result of Raz and Shpilka [RS05] by giving an efficient deterministic reconstruction algorithm for black-box noncommuting algebraic branching programs (wherein we are allowed to only query the ABP for input variables set to matrices of polynomial dimension). Our black-box model assumes that we can query for the output of *any gate* of the ABP, not just the output gate.

4.2 Noncommutative Polynomial Identity Testing

Recall that an *arithmetic circuit* C over a field $\mathbb{F}$ is defined as follows: C takes as inputs, a set of indeterminates (either commuting or noncommuting) and elements from $\mathbb{F}$ as scalars. If f, g are the inputs of an addition gate, then the output will be $f + g$. Similarly for a multiplication gate the output will be fg. For noncommuting variables the circuit respect the order of multiplication. An arithmetic circuit is a formula if the fan-out of every gate is at most one.

Noncommutative identity testing was studied by Raz and Shpilka in [RS05] and

Bogdanov and Wee in [BW05]. In the Bogdanov-Wee paper, they considered a polynomial f of small degree over $\mathbb{F}\{x_1, \cdots, x_n\}$, for a field $\mathbb{F}$, given by an arithmetic circuit. They were able to give a randomized polynomial time algorithm for the identity testing of f. The key feature of their algorithm was a reduction from noncommutative identity testing to commutative identity testing which is based on a classic theorem of Amitsur and Levitzki [AL50] about minimal identities for algebras.

Raz and Shpilka [RS05] give a deterministic polynomial-time algorithm for noncommutative formula identity testing by first converting a homogeneous formula into a noncommutative algebraic branching program (ABP), as done in [Nis91].

In this section we study the noncommutative polynomial identity testing problem. Using simple ideas from automata theory, we design a new deterministic identity test that runs in polynomial time if the input circuit is sparse and of small degree. Our algorithm works with only black-box access to the noncommuting polynomial, and we can even efficiently reconstruct the polynomial.

We will first describe the algorithm to test if a sparse polynomial of polynomial degree over noncommuting variables is identically zero. Then we give an algorithm that reconstructs this sparse polynomial. Though the latter result subsumes the former, for clarity of exposition, we describe both. Furthermore, we note that we can assume that the polynomial is given as an arithmetic circuit over a field $\mathbb{F}$.

In the case of commuting variables, [BOT88] gives an interpolation algorithm that computes the given sparse polynomial, and thus can be used for identity testing. It is not clear how to generalize this algorithm to the noncommutative setting. Our identity testing algorithm evaluates the given polynomial at specific, well-chosen points in a matrix algebra (of polynomial dimension over the base field), such that any non-zero sparse polynomial is guaranteed to evaluate to a non-zero matrix at one of these points. The reconstruction algorithm uses the above identity testing algorithm as a subroutine in a prefix-based search to find all the monomials and their coefficients.

We now describe the identity testing algorithm informally. Our idea is to view each monomial as a short binary string. A sparse polynomial, hence, is given by a polynomial number of such strings (and the coefficients of the corresponding monomials). The algorithm proceeds in two steps; in the first step, we construct a small set of finite automata such that, given any small collection of short binary strings, at least one automaton from the set accepts exactly one string from this collection; in the second step, for each of the automata constructed, we construct a tuple of points over a matrix algebra over $\mathbb{F}$ such that the evaluation of any monomial at the tuple ‘mimics’ the run of the corresponding string on the automaton. Now, given any non-zero polynomial f of small degree with few terms, we are guaranteed to have constructed an automaton A ‘isolating’ a string from the collection of strings corresponding to monomials in f. We then show that evaluating f over the tuple corresponding to A gives us a non-zero output: hence, we can conclude f is non-zero. We now describe both algorithms formally.

4.2.1 Preliminaries

We first recall some standard automata theory notation (see, for example, [HU78]). Fix a finite automaton $A = (Q, \delta, q_0, q_f)$ which takes as input strings in $\{0, 1\}^*$. Q is the set of states of A, $\delta : Q \times \{0, 1\} \to Q$ is the transition function, and q_0 and q_f are the initial and final states respectively (throughout, we only consider automata with unique accepting states). For each letter $b \in \{0, 1\}$, let $\delta_b : Q \to Q$ be the function defined by: $\delta_b(q) = \delta(q, b)$. These functions generate a submonoid of the monoid of all functions from Q to Q. This is the transition monoid of the automaton A and is well-studied in automata theory: for example, see [Str94, page 55]. We now define the 0-1 matrix $M_b \in \mathbb{F}^{|Q|\times|Q|}$ as follows:

$$M_b(q, q') = \begin{cases} 1 & \text{if } \delta_b(q) = q', \\ 0 & \text{otherwise.} \end{cases}$$

The matrix M_b is simply the adjacency matrix of the graph of the function δ_b. As the entries of M_b are only zeros and ones, we can consider M_b to be a matrix over any field $\mathbb{F}$.

Furthermore, for any $w = w_1 w_2 \cdots w_k \in \{0, 1\}^*$ we define the matrix M_w to be the matrix product $M_{w_1} M_{w_2} \cdots M_{w_k}$. If w is the empty string, define M_w to be the identity matrix of dimension $|Q| \times |Q|$. For a string w, let δ_w denote the natural extension of the transition function to w; if w is the empty string, δ_w is simply the identity function. It is easy to check that:

$$M_w(q, q') = \begin{cases} 1 & \text{if } \delta_w(q) = q', \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Thus, M_w is also a matrix of zeros and ones for any string w. Also, $M_w(q_0, q_f) = 1$ if and only if w is accepted by the automaton A.

4.2.2 The output of a circuit on an automaton

Now, we consider the ring $\mathbb{F}\{x_1, \cdots, x_n\}$ of polynomials with noncommuting variables $x_1, \cdots, x_n$ over a field $\mathbb{F}$. Let C be a noncommutative arithmetic circuit computing a polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$. Let d be an upper bound on the degree of f. We can consider monomials over the noncommuting variables $x_1, \cdots, x_n$ as strings over an alphabet of size n. For our construction in Section 4.2.3, it is convenient to encode the variables x_i in the alphabet $\{0, 1\}$. We do this by encoding the variable x_i by the string $v_i = 01^i0$, which is basically a unary encoding with delimiters. Clearly, each monomial over the x_i's of degree at most d maps uniquely to a binary string of length at most $d(n + 2)$.

Let $A = (Q, \delta, q_0, q_f)$ be a finite automaton over the alphabet $\{0, 1\}$. With respect to automaton A we have matrices $M_{v_i} \in \mathbb{F}^{|Q|\times|Q|}$ as defined in Section 4.2.1, where each v_i is the binary string that encodes x_i. We are interested in the output matrix obtained when the inputs x_i to the circuit C are replaced by the matrices M_{v_i}. This output matrix is defined in the obvious way: the inputs are $|Q|\times|Q|$ matrices and we do matrix addition and matrix multiplication at each addition (resp. multiplication) of the circuit C. We define the *output of C on the automaton A* to be this output matrix M_{out}. Clearly, given circuit C and automaton A, the matrix M_{out} can be computed in time poly$(|C|, |A|, n)$.

We observe the following property: the matrix output M_{out} of C on A is determined completely by the polynomial f computed by C; the structure of the circuit C is otherwise irrelevant. This is important for us, since we are only interested in f. In particular, the output is always 0 when $f \equiv 0$.

More specifically, consider what happens when C computes a polynomial with a single term, say $f(x_1, \cdots, x_n) = cx_{j_1} \cdots x_{j_k}$, with a non-zero coefficient $c \in \mathbb{F}$. In this case, the output matrix M_{out} is clearly the matrix $cM_{v_{j_1}} \cdots M_{v_{j_k}} = cM_w$, where $w = v_{j_1} \cdots v_{j_k}$ is the binary string representing the monomial $x_{j_1} \cdots x_{j_k}$. Thus, by Equation 4.1 above, we see that the entry $M_{out}(q_0, q_f)$ is 0 when A rejects w, and c when A accepts w. In general, suppose C computes a polynomial $f = \sum_{i=1}^{t} c_i m_i$ with t nonzero terms, where $c_i \in \mathbb{F} \setminus \{0\}$ and $m_i = \prod_{j=1}^{d_i} x_{i_j}$, where $d_i \leq d$. Let $w_i = v_{i_1} \cdots v_{i_{d_i}}$ denote the binary string representing monomial m_i. Finally, let $S_A^f = \{i \in \{1, \cdots, t\} \mid A \text{ accepts } w_i\}$.

Theorem 4.2.1 *Given any arithmetic circuit C computing polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ and any finite automaton $A = (Q, \delta, q_0, q_f)$, then the output M_{out} of C on A is such that*

$M_{out}(q_0, q_f) = \sum_{i \in S_A^f} c_i$.

Proof. The proof is an easy consequence of the definitions and the properties of the matrices M_w stated in Section 4.2.1. Note that $M_{out} = f(M_{v_1}, \cdots, M_{v_n})$. But $f(M_{v_1}, \cdots, M_{v_n}) = \sum_{i=1}^{s} c_i M_{w_i}$, where $w_i = v_{i_1} \cdots v_{i_{d_i}}$ is the binary string representing monomial m_i. By Equation 4.1, we know that $M_{w_i}(q_0, q_f)$ is 1 if w_i is accepted by A, and 0 otherwise. Adding up, we obtain the result. ■

We now explain the role of the automaton A in testing if the polynomial f computed by C is identically zero or not. Our basic idea is to try and design an automaton A that accepts exactly one word from among all the words that correspond to the non-zero terms in f. This would ensure that $M_{out}(q_0, q_f)$ is the non-zero coefficient of the monomial filtered out. More precisely, we will use the above theorem primarily in the following form, which we state as a corollary.

Corollary 4.2.2 *Given any arithmetic circuit C computing polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ and any finite automaton $A = (Q, \delta, q_0, q_f)$, then the output M_{out} of C on A satisfies:*

(1) If A rejects every string corresponding to a monomial in f, then $M_{out}(q_0, q_f) = 0$.

(2) If A accepts exactly one string corresponding to a monomial in f, then $M_{out}(q_0, q_f)$ is the nonzero coefficient of that monomial in f.

Moreover, M_{out} can be computed in time poly$(|C|, |A|, n)$.

Proof. Both points (1) and (2) are immediate consequences of the above theorem. The complexity of computing M_{out} easily follows from its definition. ■

Another interesting corollary to the above theorem is the following.

Corollary 4.2.3 *Given any arithmetic circuit C over $\mathbb{F}\{x_1, \cdots, x_n\}$, and any monomial m of degree d_m, we can compute the coefficient of m in C in time* poly$(|C|, d_m, n)$.

Proof. Apply Corollary 4.2.2 with A being any standard automaton that accepts the string corresponding to monomial m and rejects every other string. Clearly, A can be chosen so that A has a unique accepting state and $|A| = O(nd_m)$. ■

Another way to interpret the result of Corollary 4.2.3 is the following,

Corollary 4.2.4 *Given an arithmetic circuit C over $\mathbb{F}\{x_1, \cdots, x_n\}$ and a monomial m of degree d, there is an uniform way to generate a* poly$(|C|, d, n)$ *size boolean circuit C' that decides whether m is a nonzero monomial in C.*

Proof. The boolean circuit C' simply implements the algorithm described in the proof of Corollary 4.2.3. ■

Remark 4.2.5 *Observe that Corollary 4.2.3 is highly unlikely to hold in the commutative setting $\mathbb{F}[x_1, \cdots, x_n]$. For, in the commutative case, computing the coefficient of the monomial $x_1 \cdots x_n$ in even an arbitrary product of linear forms $\Pi_i \ell_i$ is at least as hard as the permanent problem over $\mathbb{F}$, which is $\#$P-complete when $\mathbb{F} = \mathbb{Q}$. To see this consider a $n \times n$ integer matrix $A = (a_{ij})_{i,j\in[n]}$. Consider the polynomial $f(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n}(\sum_{j=1}^{n} a_{ij}x_j)$. The permanent of A is the coefficient of the monomial $x_1x_2 \cdots x_n$ in f.*

Corollary 4.2.2 can also be used to give an independent proof of a weaker form of the result of Amitsur and Levitzki that is used by Bogdanov and Wee in [BW05]. We state the theorem of Amitsur and Levitzki.

Theorem 4.2.6 *[AL50] Let $M_d(\mathbb{F})$ denotes the $d \times d$ matrix algebra over the field $\mathbb{F}$. Then, $M_d(\mathbb{F})$ does not satisfy any non-trivial polynomial identity of degree $< 2d$.*

In particular, it is easy to see that the algebra $M_d(\mathbb{F})$ of $d \times d$ matrices over the field $\mathbb{F}$ does not satisfy any nontrivial identity of degree $< d$. To prove this, we will consider noncommuting monomials as strings directly over the n letter alphabet $\{x_1, \cdots, x_n\}$. Suppose $f = \sum_{i=1}^{t} c_i m_i \in \mathbb{F}\{x_1, \cdots, x_n\}$ is a nonzero polynomial of degree $< d$. Clearly, we can construct an automaton B over the alphabet $\{x_1, \cdots, x_n\}$ that accepts exactly one string, namely one nonzero monomial, say m_{i_0}, of f and rejects all the other strings over $\{x_1, \cdots, x_n\}$. Also, B can be constructed with at most d states. Now, consider the output M_{out} of any circuit computing f on B. By Corollary 4.2.2 the output matrix is non-zero, and this proves the result.

4.2.3 Construction of finite automata

We begin with a useful definition.

Definition 4.2.7 *Let W be a finite set of binary strings and $\mathcal{A}$ be a finite family of finite automata over the binary alphabet* $\{0, 1\}$.

- *We say that $\mathcal{A}$ is* isolating *for W if there exists a string $w \in W$ and an automaton $A \in \mathcal{A}$ such that A accepts w and rejects all $w' \in W \setminus \{w\}$.*
- *We say that $\mathcal{A}$ is an* (m, s)-isolating family *if for every subset $W = \{w_1, \cdots, w_s\}$ of s many binary strings, each of length at most m, there is a $A \in \mathcal{A}$ such that A is isolating for W.*

Fix parameters $m, s \in \mathbb{N}$. Our first aim is to construct an (m, s) isolating family of automata $\mathcal{A}$, where both $|\mathcal{A}|$ and the size of each automaton in $\mathcal{A}$ is polynomially bounded in size. Then, combined with Corollary 4.2.2 we will be able to obtain deterministic identity testing and interpolation algorithms in the sequel.

Recall that we only deal with finite automata that have unique accepting states. In what follows, for a string $w \in \{0, 1\}^*$, we denote by n_w the positive integer represented by the binary numeral $1w$. For each prime p and each integer $i \in \{0, \cdots, p-1\}$, we can easily construct an automaton $A_{p,i}$ that accepts exactly those w such that $n_w \equiv i \pmod{p}$. Moreover, $A_{p,i}$ can be constructed so as to have p states and exactly one final state.

Our collection of automata $\mathcal{A}$ is just the set of $A_{p,i}$ where p runs over the first few polynomially many primes, and $i \in \{0, \cdots, p-1\}$. Formally, let N denote $(m+2)\binom{s}{2}+1$; $\mathcal{A}$ is the collection of $A_{p,i}$, where p runs over the first N primes and $i \in \{0, \cdots, p-1\}$. Notice that, by the prime number theorem, all the primes chosen above are bounded in value by N^2, which is clearly polynomial in m and s. Hence, $|\mathcal{A}| = \text{poly}(m, s)$, and each $A \in \mathcal{A}$ is bounded in size by poly(m, s). In the following lemma we show that $\mathcal{A}$ is an (m, s)-isolating automata family.

Lemma 4.2.8 *The family of finite automata $\mathcal{A}$ defined as above is an (m, s)-isolating automata family.*

Proof. Consider any set of s binary strings W of length at most m each. By the construction of $\mathcal{A}$, $A_{p,i} \in \mathcal{A}$ isolates W if and only if p does not divide $n_{w_j} - n_{w_k}$ for some j and all $k \neq j$, and $n_{w_j} \equiv i \pmod{p}$. Clearly, if p satisfies the first of these conditions, i can easily be chosen so that the second condition is satisfied. We will show that there is some prime among the first N primes that does not divide

$P = \prod_{j \neq k}(n_{w_j} - n_{w_k})$. This easily follows from the fact that the number of distinct prime divisors of P is at most $\log|P|$, which is clearly bounded by $(m+2)\binom{s}{2} = N-1$. This concludes the proof. ■

We note that the above (m, s)-isolating family $\mathcal{A}$ can clearly be constructed in time poly(m, s).

4.2.4 The identity testing algorithm

We now describe the identity testing algorithm. Let C be the input circuit computing a polynomial f over $\mathbb{F}\{x_1, \cdots, x_n\}$. Let t be an upper bound on the number of monomials in f, and d be an upper bound on the degree of f. As in Section 4.2.2, we represent monomials over $x_1, \cdots, x_n$ as binary strings. Every monomial in f is represented by a string of length at most $d(n+2)$.

Our algorithm proceeds as follows: Using the construction of Section 4.2.3, we compute a family $\mathcal{A}$ of automata such that $\mathcal{A}$ is isolating for any set W with at most t strings of length at most $d(n+2)$ each. For each $A \in \mathcal{A}$, the algorithm computes the output M_{out} of C on A. If $M_{out} \neq 0$ for any A, then the algorithm concludes that the polynomial computed by the input circuit is not identically zero; otherwise, the algorithm declares that the polynomial is identically zero.

The correctness of the above algorithm is almost immediate from Corollary 4.2.2. If the polynomial is identically zero, it is easy to see that the algorithm outputs the correct answer. If the polynomial is nonzero, then by the construction of $\mathcal{A}$, we know that there exists $A \in \mathcal{A}$ such that A accepts precisely one of the strings corresponding to the monomials in f. Then, by Corollary 4.2.2, the output of C on A is nonzero. Hence, the algorithm correctly deduces that the polynomial computed is not identically zero.

As for the running time of the algorithm, it is easy to see that the family of automata $\mathcal{A}$ can be constructed in time poly(d, n, t). Also, the matrices M_{v_i} for each A (all of which are of size poly(d, n, t)) can be constructed in polynomial time. Hence, the entire algorithm runs in time poly$(|C|, d, n, t)$. We have proved the following theorem:

Theorem 4.2.9 *Given any arithmetic circuit C with the promise that C computes a polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ of degree d with at most t monomials, we can check, in time* poly$(|C|, d, n, t)$*, if f is identically zero. In particular, if f is sparse and of polynomial degree, then we have a deterministic polynomial-time algorithm.*

In the case of arbitrary noncommutative arithmetic circuits, [BW05] gives a randomized exponential time algorithm for the identity testing problem. Their algorithm is based on the Amitsur-Levitzki theorem, which forces the identity test to randomly assign exponential size matrices for the noncommuting variables since the circuit could compute an exponential degree polynomial. However, notice that Theorem 4.2.9 gives a deterministic exponential-time algorithm under the additional restriction that the input circuit computes a polynomial with at most *exponentially* many monomials. In general, a polynomial of exponential degree can have a double exponential number of terms.

4.2.5 Interpolation of noncommutative polynomials

We now describe an algorithm that efficiently computes the noncommutative polynomial given by the input circuit. Let C, f, t and d be as in Section 4.2.4. Let W denote the set of all strings corresponding to monomials with non-zero coefficients in f. For all binary strings w, let A_w denote any standard automaton that accepts w and rejects all other strings. For any automaton A and string w, we let $[A]_w$ denote the automaton that accepts those strings that are accepted by A and in addition, contain w as a prefix. For a set of finite automata $\mathcal{A}$, let $[\mathcal{A}]_w$ denote the set $\{[A]_w \mid A \in \mathcal{A}\}$.

We now describe a subroutine `Test` that takes as input an arithmetic circuit C and a set of finite automata $\mathcal{A}$ and returns a field element $\alpha \in \mathbb{F}$. The subroutine `Test` will have the following properties:

(P1) If $\mathcal{A}$ is isolating for W, the set of strings corresponding to monomials in f, then $\alpha \neq 0$.

(P2) In the special case when $|\mathcal{A}| = 1$, and the above holds, then α is in fact the coefficient of the isolated monomial.

(P3) If no $A \in \mathcal{A}$ accepts any string in W, then $\alpha = 0$.

We now give the easy description of `Test(`C`,`$\mathcal{A}$`)`:

For each $A \in \mathcal{A}$, the subroutine `Test` computes the output matrix M_{out}^A of C on A. If there is an $A \in \mathcal{A}$ such that $M_{out}^A(q_0^A, q_f^A) \neq 0$, then for the first such automaton $A \in \mathcal{A}$, `Test` returns the scalar $\alpha = M_{out}^A(q_0^A, q_f^A)$. Here, notice that q_0^A, q_f^A denote the initial and final states of the automaton A. If there is no such automaton $A \in \mathcal{A}$ is found, then the subroutine returns the scalar 0.

It follows directly from Corollary 4.2.2 that `Test` has Properties (P1)-(P3). Furthermore, clearly `Test` runs in time poly$(|C|, ||\mathcal{A}||)$, where $||\mathcal{A}||$ denotes the sum of the sizes of the automata in $\mathcal{A}$.

Let $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ denote the noncommuting polynomial computed by the input circuit C. We now describe a recursive prefix-search based algorithm `Interpolate` that takes as input the circuit C and a binary string u, and computes all those monomials of f (along with their coefficients) which contain u as a prefix when encoded as strings using our encoding $x_i \mapsto v_i = 01^i0$. Clearly, in order to obtain all monomials of f with their coefficients, it suffices to run this algorithm with $u = \epsilon$, the empty string.

In what follows, let $\mathcal{A}_0$ denote the (m, s)-isolating automata family $\{A_{p,i}\}$ as constructed in Section 4.2.3 with parameters $m = d(n+2)$ and $s = t$. As explained in Section 4.2.3, we can compute $\mathcal{A}_0$ in time poly(d, n, t).

Suppose f is the polynomial computed by the circuit C. We now describe the algorithm `Interpolate(`C`,`u`)` formally (Algorithm 1).

The correctness of this algorithm is clear from the correctness of the `Test` subroutine and Lemma 4.2.8. To bound the running time, note that the algorithm never calls `Interpolate` on a string u unless u is the prefix of some string corresponding to a monomial. Hence, the algorithm invokes `Interpolate` for at most $O(td(n+2))$ many prefixes u. Since $||[\mathcal{A}_0]_{u0}||$ and $|A_u|$ are both bounded by poly(d, n, t) for all prefixes u, it follows that the running time of the algorithm is poly$(|C|, d, n, t)$. We summarize this discussion in the following theorem.

Theorem 4.2.10 *Given any arithmetic circuit C computing a polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ of degree at most d and with at most t monomials, we can compute all the monomials of f, and their coefficients, in time* poly$(|C|, d, n, t)$. *In particular, if C computes a sparse polynomial f of polynomial degree, then f can be reconstructed in polynomial time.*

4.3 Interpolation of Algebraic Branching Programs over Noncommuting Variables

In this section, we study the interpolation problem for black-box Algebraic Branching Programs (ABP) computing a polynomial in the noncommutative ring $\mathbb{F}\{x_1, \cdots, x_n\}$. We are given as input an ABP (defined below) P in the black-box setting, and our task

Algorithm 2 The Interpolation algorithm

```
1: procedure Interpolate($C$,$u$)
2:     $\alpha, \alpha', \alpha'' \leftarrow 0$.
3:     $\alpha \leftarrow$ Test($C, \{A_u\}$)    ▷ $A_u$ is the standard automaton that accepts only $u$
4:     if $\alpha = 0$ then
5:         Break.    ▷ $u$ can not corresponds to a monomial of $f$
6:     else
7:         Output $(u, \alpha)$.    ▷ $u$ is the binary encoding of a monomial of $f$ with
   coefficient $\alpha$
8:     end if
         Now the algorithm find all monomials (along with their coefficient)
         containing $u0$ or $u1$ as prefix in the binary encoding.
9:     if $|u| = d(n+2)$ then
10:        Stop.
11:    else
12:        $\alpha' \leftarrow$Test($C, [\mathcal{A}_0]_{u0}$), $\alpha'' \leftarrow$Test($C, [\mathcal{A}_0]_{u1}$).
13:    end if
14:    if $\alpha' \neq 0$ then
15:        Interpolate($C, u0$).    ▷ There is some monomial in $C$ extending $u0$
16:    end if
17:    if $\alpha'' \neq 0$ then
18:        Interpolate($C, u1$).    ▷ There is some monomial in $C$ extending $u1$
19:    end if
20: end procedure
```

is to output an ABP P' that computes the same polynomial as P. To make the task feasible in the black-box setting, we assume that we are allowed to evaluate P at any of its intermediate gates.

We first observe that all the results in Section 4.2 hold under the assumption that the input polynomial f is allowed only *black-box access*. In the noncommutative setting, we shall assume that the black-box access allows the polynomial to be evaluated for input values from an arbitrary matrix algebra over the base field $\mathbb{F}$. It is implicit here that the cost of evaluation is polynomial in the dimension of the matrices. Note that this is a reasonable noncommutative black-box model, because if we can evaluate f only over $\mathbb{F}$ or any commutative extension of $\mathbb{F}$, then we cannot distinguish the non-commutative polynomial represented by f from the corresponding commutative polynomial. We state the black-box version of our results below.

Theorem 4.3.1 (Similar to Theorem 4.2.1) *Given black-box access to any polynomial $f = \sum_{i=1}^{t} c_i m_i \in \mathbb{F}\{x_1, \cdots, x_n\}$ and any finite automaton $A = (Q, \delta, q_0, q_f)$, then the output M_{out} of f on A is such that $M_{out}(q_0, q_f) = \sum_{i \in S_A^f} c_i$, where $S_A^f = \{i \mid 1 \leq i \leq t$ and A accepts the string w_i corresponding to $m_i\}$*

Here the output of polynomial f on A is defined analogously to the output of a circuit on A in Section 4.2.2.

Corollary 4.3.2 (Similar to Corollary 4.2.3) *Given black-box access to a polynomial f in $\mathbb{F}\{x_1, \cdots, x_n\}$, and any monomial m of degree d_m, we can compute the coefficient of m in f in time* $\text{poly}(d_m, n)$.

Finally we have,

Theorem 4.3.3 (Similar to Theorem 4.2.10) *Given black-box access to a polynomial f in $\mathbb{F}\{x_1, \cdots, x_n\}$ of degree at most d and with at most t monomials, we can compute all the monomials of f, and their coefficients, in time* $\text{poly}(d, n, t)$. *In particular, if f is a sparse polynomial of polynomial degree, then it can be reconstructed in polynomial time.*

Our interpolation algorithm for noncommutative ABPs is motivated by Raz and Shpilka's [RS05] algorithm for identity testing of ABPs over noncommuting variables. Our algorithm interpolates the given ABP layer by layer using ideas developed in Section 4.2 (principally Corollary 4.3.2).

Definition 4.3.4 *[Nis91, RS05] An Algebraic Branching Program (ABP) is a directed acyclic graph with one vertex of in-degree zero, called the source, and a vertex of out-degree zero, called the sink. The vertices of the graph are partitioned into levels numbered $0, 1, \cdots, d$. Edges may only go from level i to level $i+1$ for $i \in \{0, \cdots, d-1\}$. The source is the only vertex at level 0 and the sink is the only vertex at level d. Each edge is labeled with a homogeneous linear form in the input variables. The size of the ABP is the number of vertices.*

Notice that an ABP with no edge between two vertices u and v on levels i and $i+1$ is equivalent to an ABP with an edge from u to v labeled with the zero linear form. Thus, without loss of generality, we assume that in the given ABP there is an edge between every pair of vertices on adjacent levels.

As mentioned before, we will assume black-box access to the input ABP P where we can evaluate the polynomial computed by P at any of its gates over arbitrary matrix rings over $\mathbb{F}$. In order to specify the gate at which we want the output, we index the gates of P with a layer number and a gate number (in the layer).

Based on [RS05], we now define a *Raz-Shpilka basis* for the level i of the ABP. Let the number of nodes at the i-th level be G_i and let $\{p_1, p_2, \cdots, p_{G_i}\}$ be the polynomials computed at the nodes. We will identify this set of polynomials with the $G_i \times n^i$ matrix M_i where the columns of M_i are indexed by n^i different monomials of degree i, and the rows are indexed by the polynomials p_j. The entries of the matrix M_i are the corresponding polynomial coefficients. A Raz-Shpilka basis is a set of at most G_i linearly independent column vectors of M_i that generates the entire column space. Notice that every vector in the basis is identified by a monomial.

In the algorithm we need to compute a Raz-Shpilka basis at every level of the ABP. Notice that at the level 0 it is trivial to compute such a basis. Inductively assume we can compute such a basis at the level i. Denote the basis by $B_i = \{v_1, v_2, \cdots, v_{k_i}\}$ where $v_j \in \mathbb{F}^{G_i}$, and $k_i \leq G_i$. Assume that the elements of this basis corresponds to the monomials $\{m_1, m_2, \cdots, m_{k_i}\}$. We compute a Raz-Shpilka basis at the level $i+1$ by computing the column vectors corresponding to the set of monomials $\{m_j x_s\}_{j \in [k_i], s \in [n]}$ in M_{i+1} and then extracting the linear independent vectors out of them. Computing these column vectors requires the computation of the coefficients of these monomials, which can be done in polynomial time using the Corollary 4.3.2. Notice that we also know the monomials that the elements of this basis correspond to.

We now describe the interpolation algorithm formally. As mentioned before, we will construct the output ABP P' layer by layer such that every gate of P' computes the same polynomial as the corresponding gate in P. Clearly, this task is trivial at level 0.

Assume that we have completed the construction up to level $i < d$. We now construct level $i+1$. This only involves computation of the linear forms between level i and level $i+1$. Hence, there are $k_i \leq G_i$ vectors in the Raz-Shpilka basis at the ith level. Let the monomials corresponding to these vectors be $B = \{m_1, \cdots, m_{k_i}\}$. Fix any gate u at level $i+1$ in P, and let p_u be the polynomial compute at this gate in P. Clearly,

$$p_u = \sum_{j=1}^{G_i} p_j \ell_j$$

where p_j is the polynomial computed at the jth gate at level i, and ℓ_j is the linear form labeling the edge between the jth gate at level i and u.

We have,

$$\begin{aligned} p_u &= \sum_{j=1}^{G_i} p_j \ell_j \\ &= \sum_{j=1}^{G_i} \left(\sum_{m:|m|=i} c_m^{(j)} m \right) \left(\sum_{s=1}^{n} a_s^{(j)} x_s \right) \\ &= \sum_{m:|m|=i,s} m x_s \left(\sum_{j=1}^{G_i} c_m^{(j)} a_s^{(j)} \right) \\ &= \sum_{m:|m|=i,s} m x_s \langle c_m, a_s \rangle \end{aligned}$$

where c_m and a_s denote the vectors of field elements $(c_m^{(j)})_j$ and $(a_s^{(j)})_j$ respectively. Note that a_s denotes a vector of unknowns that we need to compute. Each monomial mx_s in the above equation gives us a linear constraint on a_s. However, this system of constraints is exponential in size. To obtain a feasible solution for $\{a_s\}_{s\in[n]}$, we observe that it is sufficient to satisfy the constraints corresponding only to monomials mx_s where $m \in B$. All other constraints are simply linear combinations of these and are thus automatically satisfied by any solution to these.

Now, for $m \in B$ and $s \in \{1, \cdots, n\}$, we compute the coefficients of mx_s in p_u and those of m in each of the p_i's using the algorithm of Corollary 4.3.2. Hence, we have all the linear constraints we need to solve for $\{a_s\}_{s\in[n]}$. Firstly, note that such a solution exists, since the linear forms in the black box ABP P give us such a solution. Moreover, any solution to this system of linear equations generates the same polynomial p_u at gate u. Hence, we can use any solution to this system of linear equations as our linear forms. We perform this computation for all gates u at the $i+1$st level. The final step in the iteration is to compute the Raz-Shpilka basis for the level $i+1$.

We can use induction on the level numbers to argue correctness of the algorithm. From the input black-box ABP P, for each level k, let $P_{jk}, 1 \leq j \leq G_k$ denote the algebraic branching programs computed by P with output gate as gate j in level k. Assume, as induction hypothesis, that the algorithm has computed linear forms for all levels upto level i and, furthermore, that the algorithm has a correct Raz-Shpilka basis for all levels upto level i. This gives us a reconstructed ABP P' upto level i with the property, for $1 \leq k \leq i$, each ABP $P'_{jk}, 1 \leq j \leq G_k$ computes the same polynomials as the corresponding $P_{jk}, 1 \leq j \leq G_k$, where P'_{jk} is obtained from P' by designating gate j at level k as output gate. Under this induction hypothesis, it is clear that our interpolation algorithm will compute a correct set of linear forms between levels i and $i+1$. Consequently, the algorithm will correctly reconstruct an ABP P' upto level $i+1$ along with a corresponding Raz-Shpilka basis for that level.

We can now summarize the result in the following theorem.

Theorem 4.3.5 *Let P be an ABP of size s and depth d over $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ given by black-box access that allows evaluation of any gate of P for inputs x_i chosen from a matrix algebra $M_k(\mathbb{F})$ for a polynomially bounded value of k. Then in deterministic time* poly(d, s, n)*, we can compute an ABP P' such that P' evaluates to the same polynomial as P.*

4.4 Noncommutative Identity Testing and Circuit Lower Bounds

In Section 4.2 we gave a new deterministic identity test for noncommuting polynomials which runs in polynomial time for sparse polynomials of polynomially bounded degree.

However, the real problem of interest is identity testing for polynomials given by small degree noncommutative circuits for which Bogdanov and Wee [BW05] give an efficient randomized test. When the noncommutative circuit is a formula, Raz and Shpilka [RS05] have shown that the problem is in deterministic polynomial time. Their method uses ideas from Nisan's lower bound technique for noncommutative formulae [Nis91].

How hard would it be to show that noncommutative PIT is in deterministic polynomial time for *circuits* of polynomial degree? In the commutative case, Impagliazzo and Kabanets [KI03] have shown that derandomizing PIT implies circuit lower bounds. It implies that either NEXP $\not\subseteq$ P/poly or the integer Permanent does not have polynomial-size arithmetic circuits.

We observe that this result also holds in the noncommutative setting. I.e., if noncommutative PIT has a deterministic polynomial-time algorithm then either NEXP $\not\subseteq$ P/poly or the *noncommutative* Permanent function does not have polynomial-size noncommutative circuits.

As noted, in some cases noncommutative circuit lower bounds are easier to prove than for commutative circuits. Nisan [Nis91] has shown exponential-size lower bounds for noncommutative formula size and further results are known for pure noncommutative circuits [Nis91, RS05]. However, proving superpolynomial size lower bounds for general noncommutative circuits computing the Permanent has remained an open problem.

The noncommutative Permanent function $Perm(x_1, \cdots, x_n) \in R\{x_1, \cdots, x_n\}$ is defined as

$$Perm(x_1, \cdots, x_n) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)},$$

where the coefficient ring R is any commutative ring with unity. Specifically, for the next theorem we consider integer permanent, i.e $R = \mathbb{Z}$.

Theorem 4.4.1 *If* PIT *for noncommutative circuits of polynomial degree* $C(x_1, \cdots, x_n) \in \mathbb{Z}\{x_1, \cdots, x_n\}$ *is in deterministic polynomial-time then either* NEXP $\not\subseteq$ P/poly *or the* noncommutative *Permanent function does not have polynomial-size noncommutative circuits.*

Proof. Suppose NEXP $\subset$ P/poly. Then, by the main result of [IKW02] we have NEXP $=$ MA. Furthermore, by Toda's theorem MA $\subseteq$ P$^{Perm_{\mathbb{Z}}}$, where the oracle computes the integer permanent. Now, assuming PIT for noncommutative circuits of polynomial degree is in deterministic subexponential-time, we will show that the (noncommutative) Permanent function does not have polynomial-size noncommutative circuits. Suppose to the contrary that it does have polynomial-size noncommutative circuits. Clearly, we can use it to compute the integer permanent as well. Furthermore, as in [KI03] we notice that the noncommutative $n \times n$ Permanent is also uniquely characterized by the identities $p_1(x) \equiv x$ and $p_i(X) = \sum_{j=1}^{i} x_{1j}p_{i-1}(X_j)$ for $1 < i \leq n$, where X is a matrix of i^2 noncommuting variables and X_j is its j-th minor w.r.t. the first row. I.e. the polynomials $p_i, 1 \leq i \leq n$ satisfy these n identities over *noncommuting* variables $x_{ij}, 1 \leq i, j \leq n$ if and only if p_i computes the $i \times i$ permanent of noncommuting variables. The rest of the proof is exactly as in Impagliazzo-Kabanets [KI03]. We can easily describe an NP machine to simulate a $\mathrm{P}^{Perm_{\mathbb{Z}}}$ computation. The NP machine guesses a polynomial-size noncommutative circuit for $Perm$ on $m \times m$ matrices, where m is a polynomial bound on the matrix size of the queries made in the computation of the $\mathrm{P}^{Perm_{\mathbb{Z}}}$ machine. Then the NP machine verifies that the circuit computes the permanent by checking the m *noncommutative* identities it must satisfy. This can be done in SUBEXP by assumption. Finally, the NP machines uses the circuit to answer all the integer permanent queries that are made in the computation of $\mathrm{P}^{Perm_{\mathbb{Z}}}$ machine. Putting it together, we get NEXP $\subseteq$ NSUBEXP which contradicts the nondeterministic time hierarchy theorem. ■

5 Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size

5.1 Introduction

We recall the Isolation Lemma from [MVV87]. Let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let U be a set of size n and $\mathcal{F} \subseteq 2^U$ be any family of subsets of U. Let $w : U \rightarrow \mathbb{Z}^+$ be a weight function that assigns positive integer weights to the elements of U. For $T \subseteq U$, define its weight $w(T)$ as $w(T) = \sum_{u \in T} w(u)$. Then, the Isolation Lemma guarantees that for any family of subsets $\mathcal{F}$ of U and for any random weight assignment $w : U \rightarrow [2n]$, with high probability there will be a unique minimum weight set in $\mathcal{F}$.

Lemma 5.1.1 (Isolation Lemma) [MVV87] Let U be an universe of size n and $\mathcal{F}$ be any family of subsets of U. Let $w : U \rightarrow [2n]$ denote a weight assignment function to elements of U. Then,

$$\text{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}] \geq \frac{1}{2},$$

where the weight function w is picked uniformly at random.

In the seminal paper [MVV87], Mulmuley et al apply the isolation lemma to give a randomized NC algorithm for computing maximum cardinality matchings for general graphs (also see [ARZ99]). Since then the isolation lemma has found several other applications. For example, it is crucially used in the proof of the result that NL $\subset$ UL/poly [RA00] and in designing randomized NC algorithms for linear representable matroid problems [NSV94]. It is also known that the isolation lemma can be used to prove the Valiant-Vazirani lemma that SAT is many-one reducible via randomized reductions to USAT.

Whether the matching problem is in deterministic NC, and whether NL $\subseteq$ UL are outstanding open problems. Thus, the question whether the isolation lemma can be derandomized is clearly important.

As noted in [Agr07], it is easy to see by a counting argument that the isolation lemma can not be derandomized, in general, because there are 2^{2^n} set systems $\mathcal{F}$. More formally, the following is observed in [Agr07].

Observation 5.1.2 [Agr07] *The Isolation Lemma can not be fully derandomized if we allow weight functions $w : U \to [n^c]$ for a constant c (i.e. weight functions with a polynomial range). More precisely, for any polynomially bounded collection of weight assignments $\{w_i\}_{i\in[n^{c_1}]}$ with weight range $[n^c]$, there exists a family $\mathcal{F}$ of $[n]$ such that for all $j \in [n^{c_1}]$, there exists two minimal weight subsets with respect to w_j.*

However that does not rule out the derandomization of any special usage of the isolation lemma. Indeed, for all applications of the isolation lemma (mentioned above, for instance) we are interested only in exponentially many set systems $\mathcal{F} \subseteq 2^U$.

We make the setting more precise by giving a general framework. Fix the universe $U = [n]$ and consider an n-input boolean circuit C where size$(C) = m$. The set 2^U of all subsets of U is in a natural 1-1 correspondence with the length n-binary strings $\{0,1\}^n$: each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$ whose i^{th} bit is 1 iff $i \in S$. Thus the n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

As an easy consequence of Lemma 5.1.1 we have the following.

Lemma 5.1.3 *Let U be an universe of size n and C be an n-input boolean circuit of size m. Let $\mathcal{F}_C \subseteq 2^U$ be the family of subsets of U defined by circuit C. Let $w : U \to [2n]$ denote a weight assignment function to elements of U. Then,*

$$\text{Prob}_w[\textit{ There exists a unique minimum weight set in } \mathcal{F}_C] \geq \frac{1}{2},$$

where the weight function w is picked uniformly at random.

Furthermore, there is a collection of weight functions $\{w_i\}_{1\leq i\leq p(m,n)}$, where $p(m,n)$ is a fixed polynomial, such that for each $\mathcal{F}_C$ there is a weight function w_i w.r.t. which there is a unique minimum weight set in $\mathcal{F}_C$.

Proof. The proof of the first part follows directly from the Isolation Lemma. To prove the second part, we use simple probabilistic argument. Suppose $w_1, w_2, \cdots, w_t$ are randomly picked weight assignment functions from $U \to [2n]$. We need to fix t suitably in the analysis. For a particular boolean circuit C of size m, the probability that none of the w_i's gives a unique minimum weight set in $\mathcal{F}_C$ is $\leq \frac{1}{2^t}$ (directly using Isolation Lemma). Notice that the number of n inputs boolean circuits of size m is $2^{O(m\log m)}$ ($m \geq n$). So the probability that none of the w_i's gives unique minimum weight $\mathcal{F}_C$ for some circuit C, is at most $\frac{2^{O(m\log m)}}{2^t}$. So the probability that for any circuit C, there exists $i \in [t]$ such that w_i gives a unique minimum weight set in $\mathcal{F}_C$ is at least $1 - \frac{2^{O(m\log m)}}{2^t}$. Choosing $t = cm\log m$ for a suitable constant $c > 0$, the above probability is nonzero. Thus we choose $p(m,n) = cm\log m$. ■

Lemma 5.1.3 allows us to formulate two natural and reasonable derandomization hypotheses for the isolation lemma.

Hypothesis 1. There is a deterministic algorithm $\mathcal{A}_1$ that takes as input (C, n), where C is an n-input boolean circuit, and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2n]$, with the property that for some w_i there is a unique minimum weight set in the set system $\mathcal{F}_C$. Furthermore, $\mathcal{A}_1$ runs in time subexponential in size(C).

Hypothesis 2. There is a deterministic algorithm $\mathcal{A}_2$ that takes as input (m, n) in unary and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2n]$, with the property that for each size m boolean circuit C with n inputs there is some weight function w_i w.r.t. which $\mathcal{F}_C$ has a unique minimum weight set. Furthermore, $\mathcal{A}_2$ runs in time polynomial in m.

Clearly, Hypothesis 2 is stronger than Hypothesis 1. It demands a "black-box" derandomization in the sense that $\mathcal{A}_2$ efficiently computes a collection of weight functions that will work for *any* set system in 2^U specified by a boolean circuit of size m.

Notice that a random collection $w_1, \cdots, w_t$ of weight functions will fulfil the required property of either hypotheses with high probability. Thus, the derandomization hypotheses are plausible. Indeed, it is not hard to see that suitable standard hardness assumptions that yield pseudorandom generators for derandomizing BPP would imply these hypotheses. We do not elaborate on this here. Our main results in this chapter are the following consequences of Hypotheses 1 and 2.

Theorem 5.1.4 *Hypothesis 1 implies that either* NEXP $\not\subset$ P/poly *or the Permanent does not have polynomial size noncommutative arithmetic circuits.*

Theorem 5.1.5 *Hypothesis 2 implies that for almost all n there is an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ in* commuting *variables x_i (where by explicit we mean that the coefficients of the polynomial f_n are computable by a uniform algorithm in time exponential in n) that does not have commutative arithmetic circuits of size $2^{o(n)}$ (where the field $\mathbb{F}$ is either the rationals or a finite field).*

The first result is a consequence of a new randomized identity testing algorithm for noncommutative circuits of small degree that is based on the isolation lemma. This algorithm is based on automata theoretic ideas introduced in Chapter 4. Then using Theorem 4.4.1, we get this result.

As a consequence of Hypothesis 2 we are able to show that for almost all n there is an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ in *commuting* variables x_i that does not have commutative arithmetic circuit of size $2^{o(n)}$. This is a fairly easy consequence of the univariate substitution idea and the observation that for arithmetic circuits computing multilinear polynomials, we can efficiently test if a given monomial has a nonzero coefficient (Lemma 5.4.1). Technically this result is similar in flavour to the Agrawal's result that a black-box derandomization of PIT for a class of arithmetic circuits via pseudorandom generators will show similar lower bound [Agr05, Lemma 5.1].

Klivans and Spielman [KS01] apply a more general form of the isolation lemma to obtain a polynomial identity test (in the commutative case). This lemma is stated below.

Lemma 5.1.6 [KS01, Lemma 4] *Let L be any collection of linear forms over variables $z_1, z_2, \cdots, z_n$ with integer coefficients in the range $\{0, 1, \cdots, K\}$. If each z_i is picked independently and uniformly at random from $\{0, 1, \cdots, 2Kn\}$ then with probability at least $1/2$ there is a unique linear form from C that attains minimum value at $(z_1, \cdots, z_n)$.*

We can formulate a restricted version of this lemma similar to Lemma 5.1.3 that will apply only to sets of linear forms L accepted by a boolean circuit C. More precisely, an integer vector $(\alpha_1, \cdots, \alpha_n)$ such that $\alpha_i \in \{0, \cdots, K\}$ is in L if and only if $(\alpha_1, \cdots, \alpha_n)$ is accepted by the boolean circuit C.

Thus, for this form of the isolation lemma we can formulate another derandomization hypothesis analogous to Hypothesis 2 as follows.

Hypothesis 3. There is a deterministic algorithm $\mathcal{A}_3$ that takes as input (m, n, K) and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2Kn]$, with the property that for any size m boolean circuit C that takes as input $(\alpha_1, \cdots, \alpha_n)$ with $\alpha_i \in \{0, \cdots, K\}$ there is some weight vector w_i for which there is a *unique* linear form $(\alpha_1, \cdots, \alpha_n)$ accepted by C which attains the minimum value $\sum_{j=1}^{n} w_i(j)\alpha_j$. Furthermore, $\mathcal{A}_3$ runs in time subexponential in size(C).

We show that Hypothesis 3 yields a lower bound consequence for the integer permanent which is similar to Impagliazzo-Kabanets result [KI03].

Remark 5.1.7 *Notice that derandomizing the isolation lemma in specific applications like the* RNC *algorithm for matchings* [MVV87] *and the containment* NL $\subseteq$ UL/poly [RA00] *might still be possible without implying such circuit size lower bounds.*

5.2 Randomized Noncommutative Identity Testing

We now describe a new randomized polynomial-time identity test for noncommutative circuits of small degree based on the isolation lemma. This is conceptually quite different from the randomized identity test of Bogdanov and Wee [BW05]. The design of our algorithm uses automata theoretic ideas described in the section 4.2.1 and section 4.2.2 of Chapter 4. In Chapter 4, we consider automata over the binary alphabet $\{0, 1\}$. In this section we will consider automata over the alphabet $\Sigma = \{x_1, x_2, \cdots, x_n\}$. It is easy to see that all the results of section 4.2.1 and section 4.2.2 of Chapter 4 are true even over an arbitrary finite alphabet and hence on $\Sigma = \{x_1, x_2, \cdots, x_n\}$.

Theorem 5.2.1 *Let $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ be a polynomial given by an arithmetic circuit C of size m. Let d be an upper bound on the degree of f. Then there is a randomized algorithm which runs in time* poly(n, m, d) *and can test whether $f \equiv 0$.*

Proof. Let $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$. Consider the set of tuples $U = [d] \times [n]$. Let $v = x_{i_1} x_{i_2} \cdots x_{i_t}$ be a nonzero monomial of f. Then the monomial can be identified with the following subset S_v of U :

$$S_v = \{(1, i_1), (2, i_2), \cdots, (t, i_t)\}.$$

Let $\mathcal{F}$ denotes the family of subsets of U corresponding to the nonzero monomials of f i.e,

$$\mathcal{F} = \{S_v \mid v \text{ is a nonzero monomial in } f\}.$$

By the Isolation Lemma, we know that if we assign random weights from $[2dn]$ to the elements of U, with probability at least $1/2$, there is a unique minimum weight set in $\mathcal{F}$. Our aim will be to construct a family of small size automata which are indexed by weights $w \in [2nd^2]$ and $t \in [d]$, such that the automaton $A_{w,t}$ will precisely accept all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w. Then from the isolation lemma we will argue that the automaton corresponding to the minimum weight will precisely accept only one string (monomial). Now for $w \in [2nd^2]$, and $t \in [d]$, we describe the construction of the automaton $A_{w,t} = (Q, \Sigma, \delta, q_0, F)$ as follows: $Q = [d] \times [2nd^2] \cup \{(0, 0)\}$, $\Sigma = \{x_1, x_2, \cdots, x_n\}$, $q_0 = \{(0, 0)\}$ and $F = \{(t, w)\}$. We define the transition function $\delta : Q \times \Sigma \to Q$,

$$\delta((i, V), x_j) = (i + 1, V + W),$$

where W is the random weight assign to $(i + 1, j)$. Our automata family $\mathcal{A}$ is simply,

$$\mathcal{A} = \{A_{w,t} \mid w \in [2nd^2], t \in [d]\}.$$

Now for each of the automata $A_{w,t} \in \mathcal{A}$, we mimic the run of the automaton $A_{w,t}$ on the circuit C as described in Section 4.2.2. If the output matrix corresponding to any of the automaton is nonzero, our algorithm declares $f \neq 0$, otherwise declares $f \equiv 0$.

The correctness of the algorithm follows easily from the Isolation Lemma. By the Isolation Lemma we know, on random assignment, a unique set S in $\mathcal{F}$ gets the minimum weight $w_{\min}$ with probability at least $1/2$. Let S corresponds to the monomial $x_{i_1} x_{i_2} \cdots x_{i_\ell}$. Then the automaton $A_{w_{\min},\ell}$ accepts the string (monomial) $x_{i_1} x_{i_2} \cdots x_{i_\ell}$. Furthermore, as no other set in $\mathcal{F}$ get the same minimum weight, $A_{w_{min},\ell}$ rejects all the other monomials. So the (q_0, q_f) entry of the output matrix M_o, that we get in running $A_{w_{\min},\ell}$ on C is nonzero. Hence with probability at least $1/2$, our algorithm correctly decides that f is nonzero. The success probability can be boosted to any constant by standard independent repetition of the same algorithm. Finally, it is trivial to see that the algorithm always decides correctly if $f \equiv 0$. ■

5.3 Proof of Theorem 5.1.4

We are now ready to prove our first result. Suppose the derandomization Hypothesis 1 holds (as stated in the introduction): i.e. suppose there is a deterministic algorithm $\mathcal{A}_1$ that takes as input (C, n) where C is an n-input boolean circuit and in subexponential time computes a set of weight functions $w_1, w_2, \cdots, w_t$, $w_i : [n] \to [2n]$ such that the set system $\mathcal{F}_C$ defined by the circuit C has a unique minimum weight set w.r.t. at least one of the weight functions w_i.

Let $C'(x_1, x_2, \cdots, x_n)$ be a noncommutative arithmetic circuit of degree d bounded by a polynomial in size(C'). By Corollary 4.2.3, there is a deterministic polynomial-time algorithm that takes as input C' and a monomial m of degree at most d and accepts if and only if the monomial m has nonzero coefficient in the polynomial computed by C'. Moreover by corollary 4.2.4, we have a uniformly generated boolean circuit C of size polynomial in size(C') that accepts only the (binary encodings of) monomials $x_{i_1} x_{i_2} \cdots x_{i_k}$, $k \leq d$ that have nonzero coefficients in the polynomial computed by C'. Now, as a consequence of Theorem 5.2.1 and its proof we have a *deterministic* subexponential algorithm for checking if $C' \equiv 0$, assuming algorithm $\mathcal{A}_1$ exists. Namely, we compute the boolean circuit C from C' in polynomial time. Then, invoking algorithm $\mathcal{A}_1$ with C as input we compute at most subexponentially many weight functions $w_1, \cdots, w_t$. Then, following the proof of Theorem 5.2.1 we construct the automata corresponding to these weight functions and evaluate C' on the matrices that each of these automata define in the prescribed manner. By assumption about algorithm $\mathcal{A}_1$, if $C' \not\equiv 0$ then one of these w_i will give matrix inputs for the variables $x_j, 1 \leq j \leq n$ on which C' evaluates to a nonzero matrix. Now the proof of the theorem follows directly from Theorem 4.4.1.

5.4 Proof of Theorem 5.1.5

In Corollary 4.2.3, we already have observed that given a noncommutative circuit C computing a small degree polynomial f and a monomial, there is an easy uniform way to check whether m is a nonzero monomial in f. Interestingly, for commutative case we can get similar result when the circuit C computes a multilinear polynomial. As this idea is useful in the proof of our main result, we state and prove it formally in the following lemma.

Corollary 5.4.1 *Given a commutative arithmetic circuit $\hat{C}$ over $\mathbb{F}[x_1, \cdots, x_n]$, with the promise that $\hat{C}$ computes a* multilinear *polynomial, and any monomial $m = \prod_{i \in S} x_i$ where $S \subseteq [n]$, we can compute the coefficient of m in C in time $\text{poly}(|\hat{C}|, n)$. Furthermore, there is an uniform way to generate a boolean circuit C' of size $\text{poly}(|C|, n)$ such that C' takes as input description of C and m and decides whether m is a nonzero monomial in C.*

Proof. Let $m = \prod_{i \in S} x_i$ be the given monomial. The algorithm will simply substitute y (a new variable) for each x_i such that $i \in S$ and 0 for each x_i such that $i \notin S$ and evaluate the circuit $\hat{C}$ to find the coefficient of the highest degree of y. The boolean circuit C' is simply the description of the above algorithm. It is clear that C' can be uniformly generated. ∎

Now we are ready to prove our main result. We will pick an appropriate multilinear polynomial $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ where:

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i,$$

where the coefficients $c_S \in \mathbb{F}$ will be determined appropriately so that the polynomial f has the claimed property.

Suppose $\mathcal{A}_2$ runs in time m^c for constant $c > 0$, where m denotes the size bound of the boolean circuit C defining set system $\mathcal{F}_C$. Notice that the number t of weight functions output by $\mathcal{A}_2$ is bounded by m^c.

The total number of coefficients c_S of f is 2^n. For each weight function w_i let $(w_{i,1}, \cdots, w_{i,n})$ be the assignments to the variables x_i. For each weight function $w_i, 1 \leq$

$i \leq t$ we write down the following equations

$$f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

Since f is of degree at most n, and the weights $w_{i,j}$ are bounded by $2n$, it is clear that $f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}})$ is a univariate polynomial of degree at most $2n^2$ in y. Thus, each of the above equations will give rise to at most $2n^2$ linear equations in the unknowns c_S.

In all, this will actually give us a system of at most $2n^2m^c$ linear equations over $\mathbb{F}$ in the unknown scalars c_S. Since the total number of distinct monomials is 2^n, and 2^n asymptotically exceeds m^c for $m = 2^{o(n)}$, the system of linear equations has a *nontrivial* solution in the c_S provided $m = 2^{o(n)}$. Furthermore, a nontrivial solution for c_S can be computed using Gaussian elimination in time exponential in n.

We claim that f does not have commutative circuits of size $2^{o(n)}$ over $\mathbb{F}$. Assume to the contrary that $\hat{C}(x_1, \cdots, x_n)$ is a circuit for $f(x_1, \cdots, x_n)$ of size $2^{o(n)}$. By Lemma 5.4.1 notice that we can uniformly construct a boolean circuit C of size $m = 2^{o(n)}$ that will take as input a monomial $\prod_{i \in S} x_i$ (encoded as an n bit boolean string representing S as a subset of $[n]$) and test if it is nonzero in $\hat{C}$ and hence in $f(x_1, \cdots, x_n)$.

Assuming Hypothesis 2, let $w_1, \cdots, w_t$ be the weight functions output by A_2 for input (m, n). By Hypothesis 2, for some weight function w_i there is a unique monomial $\prod_{j \in S} x_j$ such that $\sum_{j \in S} w_{i,j}$ takes the minimum value. Clearly, the commutative circuit $\hat{C}$ must be nonzero on substituting $y^{w_{i,j}}$ for x_j (the coefficient of $y^{\sum_{j \in S} w_{i,j}}$ will be nonzero). However, f evaluates to zero on the integer assignments prescribed by all the weight functions $w_1, \cdots, w_t$. This is a contradiction to the assumption and it completes the proof.

Remark 5.4.2 *We note that Hypothesis 2 also implies the existence of an explicit polynomial in noncommuting variables that does not have noncommutative circuits of subexponential size (we can obtain it as an easy consequence of the above proof).*

5.5 Consequence of Derandomization Hypothesis 3

We now show that the derandomization Hypothesis 3 yields a consequence exactly similar to the result of [KI03]. It particular, we get lower bound for integer permanent (under the assumption that NEXP $\not\subset$ P/poly) rather than some explicit function that we proved in Theorem 5.1.5.

Theorem 5.5.1 *If a subexponential-time algorithm $\mathcal{A}_3$ satisfying Hypothesis 3 exists then identity testing over $\mathbb{Z}$ is in* SUBEXP *which implies that either* NEXP $\not\subset$ P/poly *or the integer Permanent does not have polynomial size arithmetic circuits.*

Proof. Using Lemma 5.1.6 it is shown in [KS01, Theorem 5] that there is a randomized identity test for small degree polynomials in $\mathbb{Q}[x_1, \cdots, x_n]$, where the polynomial is given by an arithmetic circuit $\hat{C}$ of polynomially bounded degree d. The idea is to pick a random weight vector $w : [n] \to [2nd]$ and replace the indeterminate x_i by $y^{w(i)}$, where d is the total degree of the input polynomial. As the circuit $\hat{C}$ has small degree, after this univariate substitution the circuit can be evaluated in deterministic polynomial time to explicitly find the polynomial in y. By Lemma 5.1.6 it will be nonzero with probability $1/2$ if $\hat{C}$ computes a nonzero polynomial.

Coming to the proof of this theorem, if NEXP $\not\subset$ P/poly then we are done. So, suppose NEXP $\subset$ P/poly. Notice that given any monomial $x_1^{d_1} \cdots x_n^{d_n}$ of total degree bounded by d we can test if it is a nonzero monomial of $\hat{C}$ in exponential time (explicitly listing down the monomials of the polynomial computed by $\hat{C}$). Therefore, since NEXP $\subset$ P/poly there is a polynomial-size boolean circuit C that accepts the vector $(d_1, \cdots, d_n)$ iff $x_1^{d_1} \cdots x_n^{d_n}$ is a nonzero monomial in the given polynomial (as required for application of Hypothesis 3).

Now, we invoke the derandomization Hypothesis 3. We can apply the Klivans-Spielman polynomial identity test, explained above, to the arithmetic circuit $\hat{C}$ for each of the t weight vectors $w_1, \cdots, w_t$ generated by algorithm $\mathcal{A}_3$ to obtain a subexponential deterministic identity test for the circuit $\hat{C}$ by the properties of $\mathcal{A}_3$. Now, following the argument of Impagliazzo-Kabanets [KI03] it is easy to derive that the integer Permanent does not have polynomial size arithmetic circuits. ∎

Remark 5.5.2 *Although the permanent is a multilinear polynomial, notice that Hypothesis 2 does not seem strong enough to prove the above theorem. The reason is that the arithmetic circuit for the permanent that is nondeterministically guessed may not be computing multilinear polynomial and hence the application of Lemma 5.4.1 is not possible. There does not appear any easy way of testing if the guessed circuit computes a* *multilinear polynomial. The only known efficient test for multilinearity is randomized [FGL$^+$96] and that is not enough for our purpose.*

5.6 Derandomizing Valiant-Vazirani Lemma ?

Another classic result in complexity theory that is closely related to the Isolation Lemma is the well known Valiant-Vazirani Lemma. Motivated by the connection of the isolation lemma and circuit lower bound we ask similar question in the context of Valiant-Vazirani lemma, whether derandomizing similar restricted versions of the Valiant-Vazirani lemma also implies circuit lower bounds. We first recall the Valiant-Vazirani lemma as stated in the original paper [VV86].

Lemma 5.6.1 *Let $S \subseteq \{0,1\}^t$. Suppose $w_i, 1 \le i \le t$ are picked uniformly at random from $\{0,1\}^t$. For each i, let $S_i = \{v \in S \mid v.w_j = 0, 1 \le j \le i\}$ and let $p_t(S)$ be the probability that $|S_i| = 1$ for some i. Then $p_t(S) \ge 1/4$.*

Analogous to our discussion in Section 5.1, here too we can consider the restricted version where we consider $S_C \subseteq \{0,1\}^n$ to be the set of n-bit vectors accepted by a boolean circuit C of size m. We can similarly formulate derandomization hypotheses similar to Hypotheses 1 and 2.

We do not know if there is another randomized polynomial identity test for non-commutative arithmetic circuits based on the Valiant-Vazirani lemma. The automata-theoretic technique of Section 5.2 does not appear to work. Specifically, given a matrix $h : \mathbb{F}_2^n \to \mathbb{F}_2^k$, there is no deterministic finite automaton of size poly(n,k) that accepts $x \in \mathbb{F}_2^n$ if and only if $h(x) = 0$.

6 Quantum Query Complexity of Multilinear Identity Testing

6.1 Introduction

For any finite ring $(R, +, \cdot)$ the ring $R[x_1, x_2, \cdots, x_m]$ is the ring of polynomials in commuting variables $x_1, x_2, \cdots, x_m$ and coefficients in R. The ring $R\{x_1, x_2, \cdots, x_m\}$ is the ring of polynomials where the indeterminates x_i are noncommuting.

For the algorithmic problem we study in this chapter, we assume that the elements of the ring $(R, +, \cdot)$ are uniformly encoded by binary strings of length n and $R = \langle r_1, r_2, \cdots, r_k\rangle$ is given by an additive generating set $\{r_1, r_2, \cdots, r_k\}$. That is,

$$R = \{\sum_i \alpha_i r_i \mid \alpha_i \in \mathbb{Z}\}.$$

Also, the ring operations of R are performed by black-box oracles for addition and multiplication that take as input two strings encoding ring elements and output their sum or product (as defined in Chapter 2). Also, we assume that the zero element of R is encoded by a fixed string. We now define the problem which we study in this chapter.

The Multilinear Identity Testing Problem (MIT**):** The input to the problem is a black-box ring $R = \langle r_1, \cdots, r_k\rangle$ given by an additive generating set, and a multilinear polynomial $f(x_1, \cdots, x_m)$ (in the ring $R[x_1, \cdots, x_m]$ or the ring $R\{x_1, \cdots, x_m\}$) that is also given by a black-box access. The problem is to test if f is an *identity* for the ring R. More precisely, the problem is to test if $f(a_1, a_2, \cdots, a_m) = 0$ for all $a_i \in R$.

A natural example of an instance of this problem is the bivariate polynomial $f(x_1, x_2) = x_1x_2 - x_2x_1$ over the ring $R\{x_1, x_2\}$. This is an identity for R precisely when R is a commutative ring. Clearly, it suffices to check if the generators commute with each other, which gives a naive algorithm that makes $O(k^2)$ queries to the ring oracles.

Given a polynomial $f(x_1, \cdots, x_m)$ and a black-box ring R by generators, we briefly discuss some facts about the complexity of checking if $f = 0$ is an identity for R. The problem can be NP-hard when the number of indeterminates m is unbounded, even when R is a fixed ring. To see this, notice that a 3-CNF formula $F(x_1, \cdots, x_n)$ can be expressed as a $O(n)$ degree multilinear polynomial $f(x_1, x_2, \cdots, x_n)$ over $\mathbb{F}_2$, by writing F in terms of addition and multiplication over $\mathbb{F}_2$. It follows that $f = 0$ is an identity for $\mathbb{F}_2$ if and only if F is an unsatisfiable formula. However in this chapter we focus only on the upper and lower bounds on the *query complexity* of the problem.

In our query model, each ring operation, which is performed by a query to one of the ring oracles, is of unit cost. Furthermore, we consider each evaluation of $f(a_1, \cdots, a_m)$ to be of unit cost for a given input $(a_1, \cdots, a_m) \in R^m$. This model is reasonable because we consider m as a parameter that is much smaller than k.

The starting point of our study is a result of Magniez and Nayak in [MN07], where the authors study the quantum query complexity of group commutativity testing: Let G be a finite black-box group given by a generating set $g_1, g_2, \cdots, g_k$ and the group operation is performed by a group oracle. The algorithmic task is to check if G is commutative. For this problem the authors in [MN07] give a quantum algorithm with query complexity $O(k^{2/3} \log\ k)$ and time complexity $O(k^{2/3} \log^2\ k)$. Furthermore, a $\Omega(k^{2/3})$ lower bound for the quantum query complexity is also shown. The main technical tool for their upper bound result was a method of quantization of random walks first shown by Szegedy [Sze04]. More recently, Magniez et al in [MNRS07] discovered a simpler and improved description of Szegedy's method.

Our starting point is the observation that Magniez-Nayak result [MN07] for group commutativity can also be easily seen as a commutativity test for arbitrary finite black-box *rings* with similar query complexity. Furthermore, as mentioned earlier, notice that the commutativity testing for a finite ring coincides with testing if the bivariate polynomial $f(x_1, x_2) = x_1x_2 - x_2x_1$ is an identity for the ring. Since $f(x_1, x_2)$ is a multilinear polynomial, a natural question is, whether this approach would extend to testing if any multilinear polynomial is an identity for a given ring. Motivated by this connection, we study the problem of testing multilinear identities for any finite black-box ring.

The upper bound result in [MN07] is based on a group-theoretic lemma of Pak [Pak00]. Our (query complexity) upper bound result takes an analogous approach. The main technical contribution here is a suitable generalization of Pak's lemma to a multilinear polynomial setting. The multilinearity condition is crucially required. The rest of the proof is a suitable adaptation of the Magniez-Nayak result.

For the lower bound result, we show a reduction to a somewhat more general version of MIT from a problem that is closely related to the m-COLLISION problem studied in quantum computation. The m-COLLISION problem is the following. Given a function $f : \{1, 2, \cdots, k\} \to \{1, 2, \cdots, k\}$ as an oracle and a positive integer m, the task is to determine if there is some element in the range of f with exactly m pre-images.

We define the m-SPLIT COLLISION problem that is closely related to m-COLLISION problem. Here the domain $\{1, 2, \cdots, k\}$ is partitioned into m equal-sized intervals (assume k is a multiple of m) and the problem is to determine if there is some element in the range of f with exactly one pre-image in each of the m intervals. We show a reduction from m-SPLIT COLLISION to a general version of MIT. There is an easy randomized reduction from m-COLLISION problem to m-SPLIT COLLISION problem. The best known quantum query complexity lower bound for m-COLLISION problem is $\Omega(k^{\frac{2}{3}})$ [AS04] and thus we get the same lower bound for the general version of MIT that we study. Improving, the current lower bound for m-COLLISION is an important open problem in quantum computation since last few years. [1]

Our reduction for lower bound is conceptually different from the lower bound proof in [MN07]. It uses ideas from automata theory to construct a suitable black-box ring. We recently used similar ideas in the design of a deterministic polynomial-time algorithm for identity testing of noncommutative circuits computing small degree sparse polynomials [AMS08].

6.2 Black-box Rings and the Quantum Query model

We briefly explain the standard quantum query model. We modify the definition of black-box ring operations by making them unitary transformations that can be used in quantum algorithms. For a black-box ring R, we have two oracles O_R^a and O_R^m for

[1]Ambainis in [Amb07] show a quantum query complexity upper bound of $O(k^{m/m+1})$ for m-COLLISION problem.

addition and multiplication respectively. For any two ring elements r, s, and a binary string $t \in \{0,1\}^n$ we have $O_R^a|r\rangle|s\rangle = |r\rangle|r+s\rangle$ and $O_R^m|r\rangle|s\rangle|t\rangle = |r\rangle|s\rangle|rs \oplus t\rangle$, where the elements of R are encoded as strings in $\{0,1\}^n$. Notice that O_R^a is a reversible function by virtue of $(R,+)$ being an additive group. On the other hand, $(R,\cdot)$ does not have a group structure. Thus we have made O_R^m reversible by defining it as a 3-place function $O_R^m : \{0,1\}^{3n} \rightarrow \{0,1\}^{3n}$. When r or s do not encode ring elements these oracles can compute any arbitrary string.

The query model in quantum computation is a natural extension of classical query model. The basic difference is that a classical algorithm queries deterministically or randomly selected basis states, whereas a quantum algorithm can query a quantum state which is a suitably prepared superposition of basis states. Our query model closely follows the query model of Magniez-Nayak [MN07, Section 2.2]. For black-box ring operations the query operators are simply O_R^a and O_R^m (as defined above). For an arbitrary oracle function $F : X \rightarrow Y$, the corresponding unitary operator is $O_F : |g\rangle|h\rangle \rightarrow |g\rangle|h \oplus F(g)\rangle$. In the query complexity model, we charge unit cost for a single query to the oracle and all other computations are free. We will assume that the input black-box polynomial $f : R^m \rightarrow R$ is given by such an unitary operator U_f.

All the quantum registers used during the computation can be initialised to $|0\rangle$. Then a k-query algorithm for a black-box ring is a sequence of $k+1$ unitary operators and k ring oracle operators: $U_0, Q_1, U_1, \cdots, U_{k-1}, Q_k, U_k$ where $Q_i \in \{O_R^a, O_R^m, O_F\}$ are the oracle queries and U_i's are unitary operators. The final step of the algorithm is to measure designated qubits and decide according to the measurement output.

6.3 Quantum Algorithm for Multilinear Identity Testing

In this section we describe our quantum algorithm for multilinear identity testing (MIT). Our algorithm is motivated by (and based on) the group commutativity testing algorithm of Magniez and Nayak [MN07]. We briefly explain the algorithm of Magniez-Nayak. Their problem is the following: given a black-box group G by a set of generators $g_1, g_2, \cdots, g_k$, the task is to find nontrivial upper bound on the quantum query complexity to determine whether G is commutative. The group operators (corresponding to the oracle) are O_G and $O_{G^{-1}}$.

Note that for this problem, there is a trivial classical algorithm (so as quantum) of query complexity $O(k^2)$. In an interesting paper, Pak showed a classical randomized algorithm of query complexity $O(k)$ for the same problem [Pak00]. Pak's algorithm is based on the following observation ([Pak00, Lemma 1.3]): Consider a subproduct $h = g_1^{e_1} g_2^{e_2} \cdots g_k^{e_k}$ where e_i' s are picked uniformly at random from $\{0, 1\}$. Then for any proper subgroup H of G, Prob$[h \notin H] \geq 1/2$.

One important step of the algorithm in [MN07] is a generalization of Pak's lemma. Let $\mathcal{V}_\ell$ be the set of all distinct element ℓ tuples of elements from $\{1, 2, \cdots, k\}$. For $u = (u_1, \cdots, u_\ell)$, define $g_u = g_{u_1} \cdot g_{u_2} \cdots g_{u_\ell}$. Let $p = \frac{\ell(\ell-1)+(k-\ell)(k-\ell-1)}{k(k-1)}$.

Lemma 6.3.1 *[MN07] For any proper subgroup K of G,* Prob$_{u \in \mathcal{V}_\ell}[g_u \notin K] \geq \frac{1-p}{2}$.

As a simple corollary of this lemma, Magniez and Nayak show in [MN07] that, if G is non abelian then for randomly picked u and v from $\mathcal{V}_\ell$ the elements g_u and g_v will not commute with probability at least $\frac{(1-p)^2}{4}$. Thus, for non abelian G there will be at least $\frac{(1-p)^2}{4}$ fraction of noncommuting pairs (u, v). Call such pairs as *marked pairs*. Next, their idea is to do a random walk in the space of all pairs and to decide whether there exists a marked pair. They achieved this by defining a random walk and quantizing it using [Sze04]. We briefly recall the setting from [MN07, Section 2.3], and the main theorem from [Sze04], which is the central to the analysis of Magniez-Nayak result.

Quantum Walks

Let P be an irreducible and aperiodic Markov chain on a graph $G = (V, E)$ with n vertices. A walk following such a Markov chain is always ergodic and has unique stationary distribution. Let $P(u, v)$ denote the transition probability from $u \rightarrow v$, and M be a set of marked nodes of V. The goal is to make a walk on the vertices of G following the transition matrix P and decide whether M is *nonempty*. Assume that every node $v \in V$ is associated with a database $D(v)$ from which we can determine whether $v \in M$. This search procedure is modelled by a quantum walk. To analyze the performance of the search procedure, we need to consider the cost of the following operations:

Set up Cost (S): The cost to set up $D(v)$ for $v \in V$.

Update Cost (U): The cost to update $D(v)$, i.e. to update from $D(v)$ to $D(v')$, where the move $v \rightarrow v'$ is according to the transition matrix P.

Checking Cost (C): To check whether $v \in M$ using $D(v)$.

The costs are specific to the application for e.g. it can be query complexity or time complexity. The problem that we consider or the group commutativity problem of Magniez-Nayak, concern about query complexity. The following theorem due to Szegedy gives a precise analysis of the total cost involved in the quantum walk.

Theorem 6.3.2 *[Sze04] Let P be the transition matrix of an ergodic, symmetric Markov Chain on a graph $G = (V, E)$ and δ be the spectral gap of P. Also, let M be the set of all marked vertices in V and $|M|/|V| \geq \epsilon > 0$, whenever M is nonempty. Then there is a quantum algorithm which determines whether M is nonempty with constant success probability and cost $S + O((U + C)/\sqrt{\delta\epsilon})$. S is the set up cost of the quantum process, U is the update cost for one step of the walk and C is the checking cost.*

Later, Magniez-Nayak-Ronald-Santha [MNRS07] improve the total cost of the quantum walk. We state their main result.

Theorem 6.3.3 *[MNRS07] Let P be the transition matrix of a reversible, ergodic Markov Chain on a graph $G = (V, E)$ and δ be the spectral gap of P. Also let M be the set of all marked vertices in V and $|M|/|V| \geq \epsilon > 0$, whenever M is nonempty. Then there is a quantum algorithm which determines whether M is nonempty and in that case finds an element of M, with constant success probability and cost of order $S + \frac{1}{\sqrt{\epsilon}}(\frac{1}{\sqrt{\delta}}U + C)$. S is the set up cost of the quantum process, U is the update cost for one step of the walk and C is the checking cost.*

The analysis of Magniez-Nayak [MN07] is based on Theorem 6.3.2. For our problem also, we follow similar approach.

6.3.1 Query Complexity Upper Bound

Now we describe our quantum algorithm for MIT. Our main technical contribution is a suitable generalization of Pak's lemma. For any $i \in [m]$, consider the set $R_i \subseteq R$ defined as follows:

$$R_i = \{u \in R \mid \forall(b_1, \cdots, b_{i-1}, b_{i+1}, \cdots, b_m) \in R^{m-1}, f(b_1, \cdots, b_{i-1}, u, b_{i+1}, \cdots, b_m) = 0\}$$

Clearly, if f is not a zero function from $R^m \to R$, then $|R_i| < |R|$. In the following lemma, we prove that if f is not a zero function then $|R_i| \leq |R|/2$.

Lemma 6.3.4 *Let R be any finite ring and $f(x_1, x_2, \cdots, x_m)$ be a multilinear polynomial over R such that $f = 0$ is not an identity for R. For $i \in [m]$ define*

$$R_i = \{u \in R \mid \forall (b_1, \cdots, b_{i-1}, b_{i+1}, \cdots, b_m) \in R^{m-1}, f(b_1, \cdots, b_{i-1}, u, b_{i+1}, \cdots, b_m) = 0\}.$$

Then R_i is an additive coset of a proper additive subgroup of R and hence $|R_i| \le |R|/2$.

Proof. Write $f = A(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_m) + B(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_m)$ where A is the sum of all the monomials of f containing x_i and B is the sum of the rest of the monomials. Let v_1, v_2 be any two distinct elements in R_i. Then for any fixed $\bar{y} = (y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_m) \in R^{m-1}$, consider the evaluation of A and B over the points $(y_1, \cdots, y_{i-1}, v_1, y_{i+1}, \cdots, y_m)$ and $(y_1, \cdots, y_{i-1}, v_2, y_{i+1}, \cdots, y_m)$ respectively. For convenience, we abuse the notation and write,

$$A(v_1, \bar{y}) + B(\bar{y}) = A(v_2, \bar{y}) + B(\bar{y}) = 0,$$

where $\bar{y}$ is an assignment to $x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_k$ and v_1, v_2 are the assignments to x_i respectively. Note that, as f is a multilinear polynomial, the above relation in turns implies that $A(v_1 - v_2, \bar{y}) = 0$.

Consider the set $\hat{R}_i$, defined as follows: Fix any $u^{(i)} \in R_i$,

$$\hat{R}_i = \{w - u^{(i)} \mid w \in R_i\}.$$

We claim that $\hat{R}_i$ is an (additive) subgroup of R. We only need to show that $\hat{R}_i$ is closed under the addition (of R). Consider $(w_1 - u^{(i)}), (w_2 - u^{(i)}) \in \hat{R}_i$. Then $(w_1 - u^{(i)}) + (w_2 - u^{(i)}) = (w_1 + w_2 - u^{(i)}) - u^{(i)}$. It is now enough to show that for any $\bar{y} \in R^{m-1}$, $f(w_1 + w_2 - u^{(i)}, \bar{y}) = 0$ (note that $w_1 + w_2 + u^{(i)}$ is an assignment to x_i). Again using the fact that f is multilinear, we can easily see the following:

$$f(w_1 + w_2 - u^{(i)}, \bar{y}) = A(w_1, \bar{y}) + A(w_2, \bar{y}) - A(u^{(i)}, \bar{y}) + B(\bar{y})$$

and,

$$A(w_1, \bar{y}) + A(w_2, \bar{y}) - A(u^{(i)}, \bar{y}) + B(\bar{y}) = A(w_2, \bar{y}) - A(u^{(i)}, \bar{y}) = 0.$$

Note that the last equality follows because x_2 and u are in R_i. Hence we have proved that $\hat{R}_i$ is a subgroup of R. So $R_i = \hat{R}_i + u^{(i)}$ i.e. R_i is a coset of $\hat{R}_i$ inside R. Also $|R_i| < |R|$ (f is not identically zero over R). Thus, finally we get $|R_i| = |\hat{R}_i| \leq |R|/2$.

■

Our quantum algorithm is based on the algorithm of [MN07]. In the rest of the chapter, we denote by S_ℓ the set of all ℓ size *subsets* of $\{1, 2, \cdots, k\}$. We follow a quantization of a random walk on $S_\ell \times \cdots \times S_\ell = S_\ell^m$. For $u = \{u_1, u_2, \cdots, u_\ell\}$, define $r_u = r_{u_1} + \cdots + r_{u_\ell}$. Now, we suitably adapt Lemma 1 of [MN07] in our context. [2]

Let R be a finite ring given by a additive generating set $S = \{r_1, \cdots, r_k\}$. W.l.o.g. assume that r_1 is the zero element of R. Let $\hat{R}$ be a proper additive subgroup of $(R, +)$. Let j be the least integer in $[k]$ such that $r_j \notin \hat{R}$. Since $\hat{R}$ is a proper subgroup of R, such a j always exists.

Lemma 6.3.5 *Let $\hat{R} < R$ be a proper additive subgroup of R and T be an additive coset of $\hat{R}$ in R. Then* $\text{Prob}_{u \in S_\ell}[r_u \notin T] \geq \frac{1-p}{2}$, *where* $p = \frac{\ell(\ell-1)+(k-\ell)(k-\ell-1)}{k(k-1)}$.

Proof. Let j be the least integer in $[k]$ such that $r_j \notin \hat{R}$. Fix a set u of size ℓ such that $1 \in u$ and $j \notin u$. Denote by v the set obtained from u by deleting 1 and inserting j. This defines a one to one correspondence (matching) between all such pair of (u, v). Moreover $r_v = r_u + r_j$ (notice that $r_1 = 0$). Then at least one of the element r_u or r_v is not in T. For otherwise $(r_v - r_u) \in \hat{R}$ implying $r_j \in \hat{R}$, which is a contradiction.

Therefore,

$$\text{Prob}_{u \in S_\ell}[r_u \in T \mid j \in u \text{ xor } 1 \in u] \leq \frac{1}{2}.$$

For any two indices i, j,

$$\text{Prob}_{u \in S_\ell}[i, j \in u \text{ or } i, j \notin u] = \frac{\ell(\ell-1) + (k-\ell)(k-\ell-1)}{k(k-1)} = p.$$

Thus,

$$\text{Prob}_{u \in S_\ell}[r_u \in T] \leq (1-p)/2 + p \leq (1+p)/2.$$

This completes the proof. ■

Let $T = R_i$ in Lemma 6.3.5, where R_i is as defined in Lemma 6.3.4.

[2]Notice that in [MN07], the author consider the set of all ℓ tuples instead of subsets. This is important for them as they work in non abelian structure in general (where order matters). But we will be interested only over additive abelian structure of a ring and thus order does not matter for us.

Suppose $f = 0$ is not an identity for the ring R. Then, using Lemma 6.3.5, it is easy to see that, for $u_1, u_2, \cdots, u_m$ picked uniformly at random from S_ℓ, $f(r_{u_1}, \cdots, r_{u_m})$ is non zero with non-negligible probability. This is analogous to [MN07, Lemma 2]. We include a proof for the sake of completeness.

Lemma 6.3.6 *Let $f(x_1, \cdots, x_m)$ be a multilinear polynomial (in commuting or non-commuting indeterminates) over R such that $f = 0$ is not an identity for the ring R. Then,*

$$\text{Prob}_{u_1,\cdots,u_m \in S_\ell}[f(r_{u_1}, \cdots, r_{u_m}) \neq 0] \geq \left(\frac{1-p}{2}\right)^m .$$

Proof. For $i \in [m]$, let R_i be the additive coset defined in Lemma 6.3.4. The proof is by simple induction on m. The proof for the base case of the induction (i.e for $m = 1$) follows easily from the definition of R_i and Lemma 6.3.5. By induction hypothesis assume that the result holds for all t-variate multilinear polynomials g such that $g = 0$ is not an identity for R with $t \leq m - 1$.

Consider the given multilinear polynomial $f(x_1, x_2, \cdots, x_m)$. Then, by Lemma 6.3.4, R_m is a coset of an additive subgroup $\hat{R}_m$ inside R. Pick $u_m \in \mathrm{S}_\ell$ uniformly at random. If $f = 0$ is not an identity on R then by Lemma 6.3.5 we get $r_{u_m} \notin R_m$ with probability at least $\frac{1-p}{2}$. Let $g(x_1, x_2, \cdots, x_{m-1}) = f(x_1, \cdots, x_{m-1}, r_{u_m})$. Since $r_{u_m} \notin R_m$ with probability at least $\frac{1-p}{2}$, it follows that $g = 0$ is not an identity on R with probability at least $\frac{1-p}{2}$. Given that g is not an identity for R, by induction hypothesis we have that, $\text{Prob}_{u_1,\cdots,u_{m-1} \in S_\ell}[g(r_{u_1}, \cdots, r_{u_{m-1}}) \neq 0] \geq \left(\frac{1-p}{2}\right)^{m-1}$. Hence we get, $\text{Prob}_{u_1,\cdots,u_m \in S_\ell}[f(r_{u_1}, \cdots, r_{u_m}) \neq 0] \geq \left(\frac{1-p}{2}\right)^m$, which proves the lemma. ∎

We observe two simple consequences of Lemma 6.3.6. Notice that $\frac{1-p}{2} = \frac{\ell(k-\ell)}{k(k-1)}$. Letting $\ell = 1$ we get $\frac{1-p}{2} = 1/k$, and Lemma 6.3.6 implies that if $f = 0$ is not an identity for R then $f(a_1, \cdots, a_m) \neq 0$ for one of the k^m choices for the a_i from the generating set $\{r_1, \cdots, r_k\}$.

Letting $\ell = k/2$ in Lemma 6.3.6, we get $\frac{1-p}{2} \geq 1/4$. Hence we obtain the following randomized test which makes $4^m mk$ queries.

Corollary 6.3.7 *There is a randomized $4^m mk$ query algorithm for* MIT *with constant success probability, where f is m-variate and R is given by an additive generating set of size k. This can be seen as a generalization of Pak's $O(k)$ query randomized test for group commutativity.*

We use Lemma 6.3.6 to design our quantum algorithm. Technically, our quantum algorithm is similar to the one described in [MN07]. The Lemma 6.3.6 is used to guarantee that there will at least $\left(\frac{1-p}{2}\right)^m$ fraction of *marked points* in the space S_ℓ^m i.e. the points where f evaluates to non-zero. The underlying graph in our random walk is a Johnson Graph and our analysis require some simple modification of the analysis described in [MN07].

Random walk on S_ℓ

Our random walk can be described as a random walk over a graph $G = (V, E)$ defined as follows: The vertices of G are all possible ℓ subsets of $[k]$. Two vertices are connected by an edge whenever the corresponding sets differ by exactly one element. Notice that G is a connected $\ell(k-\ell)$-regular Johnson graph, with parameter $(k, \ell, \ell-1)$ [BCN89]. Let P be the normalized adjacency matrix of G with rows and columns are indexed by the subsets of $[k]$. Then $P_{XY} = 1/\ell(k-\ell)$ if $|X \cap Y| = \ell - 1$ and 0 otherwise. It is well known that the spectral gap δ of P ($\delta = 1 - \lambda$, where λ is the second largest eigenvalue of P) is $\Omega(1/\ell)$ for $\ell \leq k/2$ [BCN89]. Now we describe the random walk on G.

Let the current vertex is $u = \{u_1, u_2, \cdots, u_\ell\}$ and $r_u = r_{u_1} + r_{u_2} + \cdots + r_{u_\ell}$. With probability $1/2$ stay at u and with probability $1/2$ do the following: randomly pick $u_i \in u$ and $j \in [k] \setminus u$. Then move to vertex v such that v is obtained from u by removing u_i and inserting j. Compute r_v by simply subtracting r_{u_i} from r_u and adding r_j to it. That will only cost 2 oracle access. Staying in any vertex with probability $1/2$ ensures that the random walk is ergodic. So the stationary distribution of the random walk is always uniform. It is easy to see that the transition matrix of the random walk is $A = (I + P)/2$ where I is the identity matrix of suitable dimension. So the spectral gap of the transition matrix A is $\hat{\delta} = (1-\lambda)/2 = \delta/2$.

The query complexity analysis is similar to the analysis of Magniez-Nayak. But to fit it with our requirement, we need some careful parameter setting. We include a brief self-contained proof.

Theorem 6.3.8 *Let R be a finite black-box ring given as an oracle and $f(x_1, \cdots, x_m)$ be a multilinear polynomial over R given as a black-box. Moreover let $\{r_1, \cdots, r_k\}$ be a given additive generating set for R. Then the quantum query complexity of testing whether f is an identity for R, is $O(m(1+\alpha)^{m/2} k^{\frac{m}{m+1}})$, assuming $k \geq (1 + 1/\alpha)^{m+1}$.*

Proof. **Setup cost(S):** For the quantum walk step we need to start with an uniform distribution on S_ℓ^m. With each $u \in S_\ell$, we maintain a quantum register $|d_u\rangle$ that computes r_u. So we need to prepare the following state $|\Psi\rangle$:

$$|\Psi\rangle = \frac{1}{\sqrt{|S_\ell^m|}} \sum_{u_1,u_2,\cdots,u_m \in S_\ell^m} |u_1, r_{u_1}\rangle \otimes |u_2, r_{u_2}\rangle \otimes \cdots \otimes |u_m, r_{u_m}\rangle.$$

It is easy to see that to compute any r_{u_j}, we need $\ell - 1$ oracle access to the ring oracle. Since in each of m independent walk, quantum queries over all choices of u will be made in parallel (using quantum superposition), the total query cost for setup is $m(\ell - 1)$.

Update cost(U): It is clear from the random walk described in the section 6.3.1, that the update cost over S_ℓ is only 2 oracle access. Thus for the random walk on S_ℓ^m which is just m independent random walks, one on each copy of S_ℓ, we need a total update cost $2m$.[3]

Checking cost(C): To check whether f is zero on a point during the walk, we simply query the oracle for f once.

Recall from Szegedy's result [Sze04] (as stated in Theorem 6.3.2), the total cost for query complexity is $Q = S + \frac{1}{\sqrt{\hat{\delta}\epsilon}}(U + C)$ where $\epsilon = \left(\frac{1-p}{2}\right)^m$ is the proportion of the marked elements and $\hat{\delta}$ is the spectral gap of the transition matrix A described in section 6.3.1. Combining together we get, $Q \le m\left[(\ell-1) + \frac{3}{\sqrt{\hat{\delta}\epsilon}}\right]$. From the random walk described in the section 6.3.1, we know that $\hat{\delta} \ge \frac{1}{2\ell}$. Hence, $Q \le m\left[(\ell-1) + \frac{3\sqrt{2\ell}}{\left(\frac{1-p}{2}\right)^{\frac{m}{2}}}\right]$. Notice that, $\frac{1-p}{2} = \frac{\ell}{k}\left(\frac{1-\frac{\ell}{k}}{1-\frac{1}{k}}\right)$. Substituting for $\frac{1-p}{2}$ we get, $Q \le m\left[(\ell-1) + 3\sqrt{2}k^{m/2}\frac{1}{\ell^{\frac{m-1}{2}}\left(\frac{k-\ell}{k-1}\right)^{m/2}}\right]$. We will choose a suitably small $\alpha > 0$ so that $\frac{k-1}{k-\ell} < 1 + \alpha$. Then we can upper bound Q as follows:

$$Q \le m\left[(\ell-1) + 3\sqrt{2}\cdot(1+\alpha)^{m/2}k^{m/2}\frac{1}{\ell^{\frac{m-1}{2}}}\right].$$

Now our goal is to minimize Q with respect to ℓ and α. For that we choose $\ell = k^t$ where we will fix t appropriately in the analysis. Substituting $\ell = k^t$ we get, $Q \le m\left[(k^t - 1) + 3\sqrt{2}\cdot(1+\alpha)^{m/2}t^{1/2}k^{\frac{m-(m-1)t}{2}}\right]$. Choosing $t = (m/(m+1))$, we can

[3]In [MN07] the underlying group operation is not necessarily commutative (it is being tested for commutativity). Thus the update cost is more.

easily see that the query complexity of the algorithm is $O(m(1+\alpha)^{m/2}k^{\frac{m}{m+1}})$. Finally, recall that we need choose an $\alpha > 0$ so that $\frac{k-1}{k-\ell} \leq 1+\alpha$. Clearly, it suffices to choose α so that $(1+\alpha)\ell \leq \alpha k$. Letting $\ell = k^{m/m+1}$ we get the constraint $(1+1/\alpha)^{m+1} \leq k$ which is satisfied if $e^{(m+1)/\alpha} \leq k$. We can choose $\alpha = \frac{m+1}{\ln k}$. ■

Remark 6.3.9 *The choice of α in the above theorem shows some trade-offs in the query complexity between the parameters k and m. For constant m notice that this gives us an $O(k^{m/m+1})$ query complexity upper bound for the quantum algorithm, which is similar to the best known query upper bound for* m-COLLISION *[Amb07], when the problem instance is a function $f : [k] \to [k]$.*

Generalized Multilinear Identity Testing (GMIT**):** We now consider a variant of the MIT problem, which we call GMIT (for generalized-MIT).

Let $f : R^m \to R$ be a black-box multilinear polynomial. Consider any *additive subgroup* A of the black-box ring R, given by a set of generators $r_1, r_2, \cdots, r_k$, so that $A = \{\sum_i \beta_i r_i \mid \beta_i \in \mathbb{Z}\}$. The GMIT$(R, A, f)$ problem is the following: test whether a black-box multilinear polynomial f is an identity for A. In other words, we need to test if $f(a_1, \cdots, a_m) = 0$ for all $a_i \in A$.

It is easy to observe that the quantum algorithm actually solves GMIT and the correctness proof and analysis given in Theorem 6.3.8 also hold for GMIT problem. We summarize this observation in the following theorem.

Theorem 6.3.10 *Let R be a black-box finite ring given by ring oracles and $A = \langle r_1, r_2, \cdots, r_k \rangle$ be an* additive subgroup *of R given by generators $r_i \in R$. Let $f(x_1, x_2, \cdots, x_m)$ be a black-box multilinear polynomial $f : R^m \to R$. Then there is a quantum algorithm with query complexity $O(m(1+\alpha)^{m/2}k^{\frac{m}{m+1}})$ for the* GMIT(R, A, f) *problem (assuming $k \geq (1+1/\alpha)^{m+1}$).*

6.4 Query Complexity Lower Bound

In this section we show that GMIT problem of multilinear identity testing for additive subgroups of a black-box ring (described in Section 6.3.1), is at least as hard as m-SPLIT COLLISION (again, m-SPLIT COLLISION problem is defined in Section 6.1). Also, the well-known m-COLLISION problem can be easily reduced to m-SPLIT COLLISION problem using a simple randomized reduction. In the following lemma, we briefly state the reduction.

Lemma 6.4.1 *There is a randomized reduction from* m-COLLISION *to* m-SPLIT COLLISION *with success probability at least* e^{-m}.

Proof. Let $f : [k] \to [k]$ be a 'yes' instance of m-COLLISION, and suppose $f^{-1}(i) = \{i_1, i_2, \cdots, i_m\}$. To reduce this instance to m-SPLIT COLLISION, our idea is to pick a random m-partition $I_1, I_2, \cdots, I_m$ of the domain $[k]$ with each $|I_j| = k/m$ (for simplicity assume that m divides k). Using a standard counting argument, it is easy to see that i_j's will be placed in different blocks with high probability. In particular for a random partition $\mathcal{P}$, we show the following:

$$p = \text{Prob}_{\mathcal{P}}[\text{ For each } j \in [m], i_j's \text{ are placed in different blocks}] \geq e^{-m}.$$

Let A be the total number of possible partitions. It is easy to see that, $A = \frac{k!}{((k/m)!)^m}$.

Call a partition good if i_j's are placed in different blocks. Again, by a counting arguement, the total number of good partitions $B = \frac{m!(k-m)!}{((k/m-1)!)^m}$. Then $p = B/A = \frac{(k/m)^m}{\binom{k}{m}}$. Now using, $\binom{k}{m} \leq (\frac{ek}{m})^m$, we get $p \geq e^{-m}$. That completes the proof. ■

Consequently, showing a quantum lower bound of $\Omega(k^\alpha)$ for m-COLLISION will imply a quantum lower bound of $\Omega(k^\alpha/e^m)$ for m-SPLIT COLLISION. It will also show similar lower bound for GMIT because of our reduction.

If $f : [k] \to [k]$ is an instance of m-SPLIT COLLISION problem, then the classical randomized query complexity lower bound is $\Omega(k)$. This is observed in [MN07] for $m = 2$. Due to our reduction, we get similar randomized query complexity lower bound for GMIT.

Currently the best known quantum query complexity lower bound for m-COLLISION problem is $\Omega(k^{2/3})$ (in the case $m = 2$) [AS04]. Thus we obtain the same explicit lower bound for m-SPLIT COLLISION problem due to the random reduction from m-COLLISION to m-SPLIT COLLISION. It also implies quantum query complexity lower bound for GMIT.

Our reduction from m-SPLIT COLLISION to GMIT problem is based on automata theoretic ideas which we have used in Chapter 4 and Chapter 5.

Theorem 6.4.2 *The* m-SPLIT COLLISION *problem reduces to* GMIT *problem for additive subgroups of black-box rings.*

Proof. An instance of m-SPLIT COLLISION is a function $f : [k] \to [k]$ given as an oracle, where we assume w.l.o.g. that $k = nm$. Divide $\{1, 2, \cdots, k\}$ into m intervals $I_1, I_2, \cdots, I_m$, each containing n consecutive points of $[k]$. Recall from Section6.1 that, f is said to have an m-split collision if for some $j \in [k]$ we have $|f^{-1}(j)| = m$ and $|f^{-1}(j) \cap I_i| = 1$ for each interval I_i.

Consider the alphabet $\Sigma = \{b, c, b_1, b_2, \cdots, b_m\}$. Let A $= (Q, \Sigma, \delta, q_0, q_f)$ be a deterministic finite state automaton that accepts all strings $w \in \Sigma^*$ such that each $b_j, 1 \leq j \leq m$ occurs at least once in w. It is easy to see that such an automaton with a single final state q_f can be designed with total number of states $|Q| = 2^{O(m)} = t$. W.l.o.g. let the set of states Q be renamed as $\{1, 2, \cdots, t\}$, where 1 is the initial state and t is the final state.

For each letter $a \in \Sigma$, let M_a denote the $t \times t$ transition matrix for δ_a (as defined in Section 4.2.1 and used in Section 5.2 for any arbitrary alphabet Σ). Since each M_a is a $t \times t$ 0-1 matrix, each M_a is in the ring $\mathrm{M}_t(\mathbb{F}_2)$ of $t \times t$ matrices with entries from the field $\mathbb{F}_2$. Let R denote the k-fold product ring $(\mathrm{M}_t(\mathbb{F}_2))^k$. Clearly, R is a finite ring (which is going to play the role of the black-box ring in our reduction). We now define an additive subgroup T of R, where we describe the generating set of T using the m-SPLIT COLLISION instance f.

For each index $i \in [k]$, define an k-tuple $T_i \in R$ as follows. If $i \neq f(i)$, then define $T_i[i] = M_b$, $T_i[f(i)] = M_{b_j}$ (where $i \in I_j$) and and for each index $s \notin \{i, f(i)\}$ define $T_i[s] = M_c$. For $i = f(i)$, define $T[i] = M_{b_j}$ $(i \in I_j)$ and the rest of the entries as M_c. The additive subgroup of R that we consider is $T = \langle T_1, T_2, \cdots, T_k \rangle$ generated by the $T_i, 1 \leq i \leq k$.

Furthermore, define two $t \times t$ matrices A and B in $\mathrm{M}_t(\mathbb{F}_2)$ as follows. Let $A[1, 1] = 1$ and $A[u, \ell] = 0$ for $(u, \ell) \neq (1, 1)$. For the matrix B, let $B[t, 1] = 1$ and $B[u, \ell] = 0$ for $(u, \ell) \neq (t, 1)$.

Claim 6.4.3 *Let $w = w_1 w_2 \cdots w_s \in \Sigma^*$ be any string. Then the automaton* A *defined above accepts w if and only if the matrix $AM_{w_1} M_{w_2} \cdots M_{w_s} B$ is nonzero.*

Proof of Claim: By definition of the matrices M_a, the $(1, t)^{th}$ entry of the product $M_{w_1} M_{w_2} \cdots M_{w_s}$ is 1 if and only if w is accepted by A. By definition of the matrices A and B the claim follows immediately.

Now, consider the polynomial $P(x_1, x_2, \cdots, x_m)$ with coefficients from the matrix ring R defined as follows:

$$P(x_1, x_2, \cdots, x_m) = \bar{A} x_1 x_2 \cdots x_m \bar{B},$$

where $\bar{A} = (A, A, \ldots, A) \in R$ and $\bar{B} = (B, B, \cdots, B) \in R$ are k-tuples of A's and B's respectively. We claim that the multilinear polynomial $P(x_1, x_2, \cdots, x_m) = 0$ is an identity for the additive subgroup T if and only if f has no m-split collision.

Claim 6.4.4 $P(x_1, \cdots, x_m) = 0$ *is an identity for the additive subgroup* $T = \langle T_1, \cdots, T_k \rangle$ *if and only if f has no m-split collision. In other words,* GMIT(R, T, P) *is an 'yes' instance if and only if f has no m-split collision.*

Proof of Claim: Suppose f has an m-split collision. Specifically, let $i_j \in I_j$ ($1 \leq j \leq m$ and $i_1 < i_2 < \cdots < i_m$) be indices such that $f(i_1) = \cdots = f(i_m) = \ell$. In the polynomial P, we substitute the indeterminate x_j by T_{i_j}.

Then $P(T_{i_1}, T_{i_2}, \cdots, T_{i_m}) = \bar{A} M \bar{B}$, where $M = T_{i_1} \cdots T_{i_m}$. M is a k-tuple of $t \times t$ matrices such that the ℓ^{th} component of M is $\prod_{j=1}^{m} M_{b_j}$ where $i_j \in I_j$. Since $b_{i_1} b_{i_2} \cdots b_{i_m} \in \Sigma^*$ is a length m-string containing all the b_j's it will be accepted by the automaton A. Consequently, the $(q_0, q_f)^{th}$ entry of the matrix M, which is the $(1, t)^{th}$ entry, is 1 (as explained in Section 4.2.1). It follows that the $(1, 1)$ entry of the matrix AMB is 1. Hence $P = 0$ is not an identity over the additive subgroup T.

For the other direction, assume that f has no m-split collision. We need to show that $P = 0$ is an identity for the ring T. For any m elements $S_1, S_2, \cdots, S_m \in T$ consider $P(S_1, S_2, \cdots, S_m) = \bar{A} S_1 S_2 \cdots S_m \bar{B}$. Since Each S_j is an $\mathbb{F}_2$-linear combination of the generators $T_1, \cdots, T_k$, it follows by distributivity in the ring R that $P(S_1, S_2, \cdots, S_m)$ is an $\mathbb{F}_2$-linear combination of terms of the form $P(T_{k_1}, T_{k_2}, \cdots, T_{k_m})$ for some m indices $k_1, \cdots, k_m \in [k]$. Thus, it suffices to show that $P(T_{k_1}, T_{k_2}, \cdots, T_{k_m}) = 0$.

Let $\hat{T} = T_{k_1} T_{k_2} \cdots T_{k_m}$. Then, for each $j \in [k]$ we have $\hat{T}[j] = T_{k_1}[j] T_{k_2}[j] \cdots T_{k_m}[j]$. Since f has no m-split collision, for each $j \in [N]$ the set of matrices $\{M_{b_1}, M_{b_2}, \cdots, M_{b_m}\}$ is *not* contained in the set $\{T_1[j], T_2[j], \cdots, T_k[j]\}$. Thus, $\hat{T}[j] = T_{k_1}[j] T_{k_2}[j] \cdots T_{k_m}[j]$ is a product of matrices $M_{w_1} M_{w_2} \cdots M_{w_m}$ for a word $w = w_1 w_2 \cdots w_m$ that is not accepted by A. It follows from the previous claim that $A \hat{T}[j] B = 0$. Hence we have, $P(T_{k_1}, T_{k_2}, \cdots, T_{k_m}) = 0$, which completes the proof. ■

In Section 6.3.1, we have already shown a quantum algorithm of query complexity $O(k^{\frac{m}{m+1}})$ for MIT (m is a constant). This bound holds as well for GMIT. We conclude this section by showing that any algorithm of query complexity $q(k,m)$ (q is any function) for GMIT will give an algorithm of similar query complexity for m-COLLISION problem. In particular an algorithm for GMIT of query complexity $k^{o(m/m+1)}$ will improve the best known algorithm for m-COLLISION problem due to Ambainis [Amb07]. The following corollary is an easy consequence of Theorem 6.4.2.

Corollary 6.4.5 *Let* $f : [k] \rightarrow [k]$ *be an instance of* m-SPLIT COLLISION *problem and* GMIT(R,T,P) *be an instance of* GMIT *problem, where the multilinear polynomial* $P : R^m \rightarrow R$ *and* T *is an additive subgroup of* G *given by* k *generators. Then, if we have a quantum algorithm of query complexity* $q(k,m)$ *for* GMIT *problem, we will have a quantum algorithm for* m-SPLIT COLLISION *with query complexity* $O(q(k,m))$.

Proof. Let $\mathcal{A}$ be an algorithm for GMIT with quantum query complexity $q(k,m)$. Given an instance of m-SPLIT COLLISION, the generators for the additive subgroup T is indexed by $1,2,\cdots,k$ (as defined in the proof of Theorem 6.4.2). Also, define the polynomial $P(x_1,x_2,\cdots,x_m)$ So the inputs of our GMIT problem are $1,2,\cdots,k$ and P. Using the algorithm $\mathcal{A}$, we define another algorithm $\mathcal{A}'$ which does the following. When $i \in [k]$ is invoked by $\mathcal{A}$ for the ring operation, the algorithm $\mathcal{A}'$ constructs the generator T_i by making only one query to the oracle for f. One more query to the f-oracle is required to erase the output. Moreover, if $\mathcal{A}$ wants to check whether the output of the ring operation is a valid generator (say T_j for some j), then also $\mathcal{A}'$ uses just two queries to the oracle of f. Thus we have an algorithm $\mathcal{A}'$ for m-SPLIT COLLISION with query complexity $4q(k)$. ■

Recall that the best known lower bound for m-SPLIT COLLISION problem is $\Omega(k^{2/3})$. Then, combining Theorem 6.4.2 and Corollary 6.4.5, we get $\Omega(k^{2/3})$ quantum query lower bound for GMIT problem.

Bibliography

[AB03] Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. *J. ACM*, 50(4):429–443, 2003.

[ADM06] Vikraman Arvind, Bireswar Das, and Partha Mukhopadhyay. The complexity of black-box ring problems. In *COCOON*, pages 126–135, 2006.

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS*, pages 92–105, 2005.

[Agr07] Manindra Agrawal. Rings and integer lattices in computer science. *Barbados Workshop on Computational Complexity*, Lecture no 9, 2007.

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: depth reduction and size lower bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[AL50] S.A Amitsur and J. Levitzki. Minimal identities for algebras. *In Proceedings of the American Mathematical Society*, 1:449–463, 1950.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AM69] Michael F. Atiyah and Ian G. Macdonald. Introduction to commutative algebra. *Addison-Wesley Publishing Company*, 1969.

[AM07] Vikraman Arvind and Partha Mukhopadhyay. The monomial ideal membership problem and polynomial identity testing. In *ISAAC, (Full Version: ECCC report TR07-095),* pages 800–811, 2007.

[AM08] Vikraman Arvind and Partha Mukhopadhyay. Derandomizing the isolation lemma and lower bounds for circuit size. In *APPROX-RANDOM,* pages 276–289, 2008.

[AM09] Vikraman Arvind and Partha Mukhopadhyay. Quantum query complexity of multilinear identity testing. In *STACS'09 (To appear)*, 2009.

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM J. Comput.*, 37(1):210–239, 2007.

[AMS08] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on noncommutative and commutative polynomial identity testing. In *IEEE Conference on Computational Complexity*, pages 268–279, 2008.

[ARZ99] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting uniform and nonuniform upper bounds. *J. Comput. Syst. Sci.*, 59(2):164–181, 1999.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, 1998.

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.

[AS05] Manindra Agrawal and Nitin Saxena. Automorphisms of finite rings and applications to complexity of problems. In *STACS'05, Springer LNCS 3404*, pages 1–17, 2005.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *FOCS*, 2008.

[Bab91] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 164–174, 1991.

[Bab92] László Babai. Bounded round interactive proofs in finite groups. *SIAM J. Discret. Math.*, 5(1):88–111, 1992.

[BCN89] A.E. Brouwer, A.M. Cohen, and A.N. Neumaier. Distance Regular Graphs. *Springer-Verlag*, pages 255–256, 1989.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A. Shokrollahi. Algebraic Complexity Theory. *Springer-Verlag, Series: Grundlehren der mathematischen Wissenschaften*, 315, 1997.

[Ber67] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell Systems Technical Journal*, 46:1853–1859, 1967.

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proc. of the 20th annual ACM Sym. on Theory of computing*, pages 301–309, 1988.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[BS84] László Babai and Endre. Szemerédi. On the complexity of matrix group problems I. *25th IEEE Symposium of Foundations of Computer Science*, 24-26:229 – 240, 1984.

[BW05] Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing. In *CCC '05: Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 92–99, 2005.

[CK00] Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. *SIAM J. Comput.*, 29(4):1247–1256, 2000.

[CLO92] David Cox, John Little, and Donal O'Shea. Ideals, Varieties and Algorithms. *Undergraduate Text in Mathematics, Springer*, 1992.

[CM01] Kevin K.H. Cheung and Michele Mosca. Decomposing finite abelian groups. *Quantum Information and Computation*, 1(2):26–32, 2001.

[CS07] Steve Chien and Alistair Sinclair. Algebras with polynomial identities and computing the determinant. *SIAM J. Comput.*, 37(1):252–266, 2007.

[DS06] Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2006.

[FGL$^+$96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[GK98] Dima Grigoriev and Marek Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *STOC ’98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 577–582, 1998.

[GR98] Dima Grigoriev and Alexander Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. In *FOCS ’98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 269, 1998.

[GR05] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. In *FOCS ’05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 407–418, 2005.

[Her26] G. Hermann. Die frage der endlich viel schritte in der theorie der polynomideale. *Math. Annalen, 95*, pages 736–788, 1926.

[HU78] John E. Hopcroft and Jeffrey D. Ullman. Introduction to Automata Theory, Languages and Computation. *Addison-Wesley*, 1978.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[IMS03] Gábor Ivanyos, Frédéric Magniez, and Miklos Santha. Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem. *Int. J. Found. Comput. Sci.*, 14(5):723–740, 2003.

[KB79] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979.

[KI03] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC ’03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 355–364, 2003.

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223, 2001.

[KS06] Neeraj Kayal and Nitin Saxena. Complexity of ring morphism problems. *Comput. Complex.*, 15(4):342–390, 2006.

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Comput. Complex.*, 16(2):115–138, 2007.

[KS08] Zohar Karnin and Amir Shpilka. Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual CCC*, pages 280–291, 2008.

[Len92] Hendrik W. Lenstra. Algorithms in algebraic number theory. *Bulletin of the AMS, 26(2)*, pages 211–244, 1992.

[LFK92] Carsten Lund, Lance Fortnow, and Howard Karloff. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Lov79] László Lovasz. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory : Proc. of the Conference on Algebraic, Arithmetic, and Categorical Methods in Computation Theory, Akademic-Verlag, Vol.2*, pages 565–574, 1979.

[LV98] Daniel Lewin and Salil Vadhan. Checking polynomial identities over any field: towards a derandomization? In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 438–447, 1998.

[May89] E. Mayr. Membership in polynomial ideals over q is exponential space complete. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science on STACS 89*, pages 400–406, 1989.

[McD74] Bernard R. McDonald. Finite rings with identity. *Marcel Dekker, INC. New York*, 1974.

[MM82] E.W. Mayr and A.R. Meyer. The complexity of word problem for commutative semigroups and polynomial ideals. *Adv. Math. 46*, pages 305–329, 1982.

[MN07] Frederic Magniez and Ashwin Nayak. Quantum complexity of testing group commutativity. *Algorithmica*, 48(3):221–232, 2007.

[MNRS07] Frederic Magniez, Ashwin Nayak, Jeremie Roland, and Miklos Santha. Search via quantum walk. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 575–584, 2007.

[Mos99] Michele Mosca. Quantum Computer Algorithms. PhD Thesis. *University of Oxford*, 1999.

[MR01] Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithm. *Cambridge*, 2001.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354, 1987.

[NC00] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. *Cambridge*, pages 240–242, 2000.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 410–418, 1991.

[NSV94] H. Narayanan, Huzur Saran, and Vijay V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences and edge-disjoint spanning trees. *SIAM J. Comput.*, 23(2):387–397, 1994.

[Pak00] Igor Pak. Testing commutativity of a group and the power of randomization. *Electronic Version at: http://www-math.mit.edu/pak/research.html*, 2000.

[RA00] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000.

[Raz04] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. In *STOC*, pages 633–641, 2004.

[Raz06] Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing*, 2(1):121–135, 2006.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RSY07] Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. In *FOCS*, pages 438–448, 2007.

[RY08] Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. In *IEEE Conference on Computational Complexity*, pages 128–139, 2008.

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP (1)*, pages 60–71, 2008.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sch98] Alexander Schrijver. Theory of linear and integer programming. *John Wiley*, pages 45–59, 1998.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Str73a] Volker Strassen. Die berechnungskomplexiät von elementarsymmetrischen funktionen und von interpolationskoeffizienten. *Numer. Math*, 20:238–251, 1973.

[Str73b] Volker Strassen. Vermeidung von divisionen. *Journal of Reine Angew. Math.*, 264:182–202, 1973.

[Str94] Howard Straubing. Finite Automata, Formal Logic, and Circuit Complexity. *Progress in Theoretical Computer Science, Birkhäuser*, 1994.

[Sud98] Madhu Sudan. Lectures on algebra and computation (6.966). *Electronic Version at:http://people.csail.mit.edu/madhu/FT98/course.html, Lecture 12,13,14*, 1998.

[SV08] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 507–516, 2008.

[SW01] Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001.

[Sze04] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41, 2004.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. Np is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(1):85–93, 1986.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. Modern Computer Algebra. *Cambridge University Press, 2nd Ed.*, 2003.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM '79: Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, pages 216–226, 1979.