Lower Bounds Related to Polynomial Identity Testing, Circuit Minimization, and Inversion Minimization

By

Andrew Morgan

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2022

Date of final oral examination: 01/21/2022

The dissertation is approved by the following members of the Final Oral Committee:

Dieter van Melkebeek, Professor, Computer Sciences
Eric Allender, Professor, Computer Science at Rutgers University
Jin-Yi Cai, Professor, Computer Sciences
Daniel Erman, Associate Professor, Mathematics

Contents

Abstract

Computational complexity theory aims to provide a mathematically rigorous understanding of the resources necessary to solve computational problems. This dissertation develops complexity lower bounds in three settings.

Polynomial Identity Testing. Given two arithmetic formulas, do they compute the same multivariate polynomial? This basic problem has an efficient randomized algorithm and plays a central role in the area of derandomization. We propose a pseudorandom generator based on evaluations of univariate rational functions and characterize its power through its vanishing ideal. We construct an explicit Groebner basis, yielding tight bounds on the minimum sparsity, degree, and partition class size of set multilinearity in the vanishing ideal. We develop a membership test inspired by the theory of alternating algebras and give a proof of concept of its usefulness in the model of read-once oblivious algebraic branching programs. Via an equivalence with the Shpilka–Volkovich generator, computational lower bounds for polynomials in the vanishing ideal follow from existing derandomization results.

Circuit Minimization. Given a truth table, what is the minimum size of a Boolean circuit computing it? This fundamental problem and a descriptive Kolmogorov variant known as MKTP have thus far resisted classification as solvable in polynomial time or NP-complete. We show that the complexity of MKTP is no less than that of isomorphism problems with respect to zero-error randomized reductions. Our reduction is related to

the known interactive proofs of nonisomorphism and hinges on a near-optimal encoding for samples of flat, samplable distributions.

Inversion Minimization. Given a rooted tree and a ranking of its leaves, what is the minimum number of inversions of the leaves that can be attained by ordering the tree? We study the number of comparisons necessary to solve this problem by considering notions of connectivity and sensitivity with respect to adjacent-rank transpositions. We show that for many tree shapes the problem is essentially as hard as sorting.

Acknowledgements

This dissertation would not have been possible without the tremendous collaboration, mentorship, education, support, friendship, and love that I have received throughout my life. The diversity and magnitude of these contributions make it impossible to memorialize all of them properly in such a small space. If your name is below, know that it is only a small token for the gratitude I feel. To those inadvertently left out, I can at least address you first: sorry and thank you!

To my advisor, Dieter van Melkebeek: Appropriately acknowledging what you have done for this dissertation—let alone for my personal and professional development in general—would probably require at least a chapter of its own. You have guided my research, painstakingly tutored my writing, advocated for my success, supported and encouraged me through droughts, mentored my teaching, and altogether demonstrated yourself to be a paragon of dedication and competence. From your example I have learned not only how to conduct and present research, but also how to teach, how to work, and how to think. While I still have much to learn, it is both a great comfort and a terrific privilege to have such a strong example to follow. Thank you!

To my other coauthors, Eric Allender, Joshua Grochow, and Cristopher Moore: I deeply appreciate having had the opportunity to collaborate with you. The paper presenting the results in Chapter 3 was my first publication and cut short a growing anxiety that I might never have one. To my colleagues Marco Carmosino, Shuichi Hirahara, Or Meir, Gautam Prakriya, and Joseph Swernofsky: I thank you for sharing and discussing your ideas with

me and for listening to my own. To my committee members Jin-Yi Cai and Daniel Erman: Thank you for taking the time to see me through the last steps.

To the UW–Madison Computer Sciences department, Cisco Systems, and the National Science Foundation: Thank you for providing funding during my time in graduate school. To the Simons Institute: Thank you for allowing me to visit for a semester. To the Vilas Trust: Thank you for providing travel grants.

To my family—my parents, Gail and Kevin Morgan; my sibling, Erica Morgan; my wife, Katie Morgan; my in-laws, Bernie and Bernie Windschitl, Jessie and Jayce Dowell, and Sarah, Matt, Ian, Lily, and Lizzy Dahl; and my cats, Jet and Leena—you have been a major source of support through my time in graduate school. Thank you, and I love you all!

To my friends past and present—Adrien, Alexi, Alisa, Ben, Brendan, Bryce B., Bryce S., Dana, Darrin, Diana, Hilary, Jared, Josh, Katie L., Katie S., Kayla, Kevin K., Liz, Marc, Menghui, Rhea, Romin, Shams, Smitty, Steve, Tim, Tyler, and Zach—I would not be where I am today without you all. Thank you for the good times!

To all of my teachers, mentors, and role models—in particular the many already mentioned as well as Uri Andrews, Eric Bach, Jim Barnette, Nigel Boston, Mingzhong Cai, Curt Canaday, Keith Chapple, Shuchi Chawla, Andrzej Czygrinow, Jacob Davis, Liz Floerchinger, Jeff Hall, Robert Harron, Suzanne Fullerton, Lalit Jain, Bill Janssen, Zev Klagsbrun, Katie Kolossa, Bill Mellyn, Brian Rice, Brad Ross, Michele Schwanenberger, Andreas Seeger, Michael Shunk, Hal Smith, John Ulbright, and Randy Vogl—your passion and support have been inspirational and greatly appreciated. I hope this work makes you proud!

Finally, to Laura Bailey, Taliesin Jaffe, Ashley Johnson, Matthew Mercer, Liam O'Brien, Marisha Ray, Sam Riegel, Travis Willingham, and the many others involved along the way: Thank you for creating the masterclass in collaborative storytelling that has helped keep me sane, happy, and grounded the past few years—Critical Role!

Chapter 1

Introduction

One of the most marvelous principles that every computer programmer learns is that *simple instructions can describe complex behavior*. Computers are, at their most fundamental level, simple: All data are represented as bits, there is a small set of simple instructions to operate on those bits, and programs are just sequences of instructions from that set. The art of computer programming lies in the arrangement of the simple instructions so that the effect is interesting or useful. By arranging the simple instructions into complex and capable algorithms, then arranging those algorithms into even more complex and capable algorithms, and so on, programmers have created all the sophisticated software we know.

Part of the marvel in this principle is the empowerment that it entails. Through programming, we can express how to do tedious work in the simple instructions of a computer, after which the computer will perform the work for us, usually much faster, with fewer errors, and while expending less energy. Expressing complex processes as computer instructions has been a major throughline in humanity's recent great accomplishments: setting foot on the moon, sequencing genomes and producing vaccines, rendering life-like animations, as well as connecting family, friends, and colleagues in real-time across disparate parts of the globe.

There is also the breadth of the principle: It applies to more than electronic computers. From the construction of ancient wonders to the direction of armies in war, we see enor-

mously complex undertakings reduced to instructions for laymen. According to free-market economic theory, the comparatively complex outcome of social welfare follows from every individual simply acting in their own self-interest. More fundamentally, Newton's laws of motion describe our everyday physics with profound concision. Even at scales where Newton's laws break down, quantum mechanics and general relativity have predictive power far surpassing their mathematical sophistication. Arguably, even human cognition is itself an emergent behavior determined by the physics of the matter comprising the brain.

The power and breadth of the principle even lead some computer programmers to internalize its converse, that *every* complex behavior could be reduced to simple instructions. Alas, in both theory and practice, there are limits. Solving the halting problem is a complex behavior that, provably, no algorithm can exhibit. Factoring integers is a complex behavior for which there are algorithms, but, even for integers with only a few hundred digits, all the algorithms that we are physically capable of running require hundreds of thousands of years to complete. It is not known with mathematical certainty that a faster algorithm does not exist, but such an algorithm has eluded centuries of search by mathematicians and computer scientists.

This leads us to consider the following questions, the guiding questions of this thesis: What are the limits of algorithms? For a given complex behavior, can it be reduced to simple instructions, and, more specifically, can it be realized without tremendous expense of resources? Providing mathematically rigorous answers to these and related questions is the quest of computational complexity theory.

Lower Bounds One way to answer the guiding question is to prove a *lower bound*. In a lower bound, one establishes a model of computation that formalizes the simple instructions and means of combining them into algorithms. Associated to the model are various resources, such as how much time or memory space is necessary to execute an algorithm. The desired complex behavior is then expressed as a formal specification, typically a mapping from input

to outputs, for what the algorithm should do in the terms of the model. A lower bound states that every algorithm in the model that behaves according to the specification requires *at least* a certain amount of resources.

A lower bound constitutes a direct answer to the guiding question for the specific model of computation and complex behavior studied. In addition to this intrinsic appeal, lower bounds play key roles in other questions in computational complexity theory. A major example concerns the utility of randomness as a computational resource: How useful are coin flips? The answer to this question turns out to be equivalent to certain lower bounds. In a similar vein, the mathematical security of cryptography rests on other lower bounds. In both settings, the relevant lower bounds remain unproven.

The relationship between lower bounds and randomness as a computational resource plays a significant role in some of the work in this thesis. The principle underlying the relationship is that *whether something looks random to you depends on your computational resources*. When a computation uses randomness and a limited amount of another resource, the idea is to replace the randomness by *pseudorandomness*. Pseudorandomness differs from true randomness in that it is constructed in a deterministic fashion from a comparatively small seed of random bits; nevertheless, it *looks* like true randomness to any computation with the resource limitation. Necessary for the effect is a lower bound, namely that no computation with the resource limitation can distinguish the pseudorandomness from true randomness. Conversely, it is possible to translate a variety of lower bounds into pseudorandomness.

This thesis presents three units of lower bounds. They relate to *polynomial identity testing* in an algebraic model, *minimizing circuits* in a universal model of computation, and *minimizing inversions* in a comparison-based model.

1.1 Polynomial Identity Testing

The first unit concerns lower bounds related to the topic of *polynomial identity testing*. Here the primary model of computation is algebraic: Inputs are scalars from a field (such as the rational numbers), and computation consists of combining those inputs using addition, subtraction, and multiplication. If we represent the inputs by variables, say x, y, z, then computations are formulas that express polynomials in those variables. For example, the formula

$$y \cdot (x - 2z) - 2xz - yz \tag{1.1}$$

expresses a polynomial using three additions/subtractions and five multiplications. A second formula, this time with only two additions/subtractions and four multiplications, is the following:

$$x \cdot (y - 2z) - 3yz. \tag{1.2}$$

As it turns out, (1.1) and (1.2), despite representing distinct computations, nevertheless yield the same polynomial. To see this, we can use the distributive law to expand each as a sum of monomials. The result in both cases is

$$xy - 2xz - 3yz. \tag{1.3}$$

Polynomial identity testing is the general problem of deciding whether two formulas compute the same polynomial. One way to solve the problem is as above: Expand both formulas as a sum of monomials, and compare the coefficients of each monomial. This algorithm can be quite slow in general, because the expansion can be much larger than the size of the formula. For example, $(1 + x_1)(1 + x_2)\cdots(1 + x_{100})$ can be written on a single sheet of paper, but there are more monomials in its expansion than there are atoms in a single-sided printing of this dissertation.

A much faster—and even simpler—algorithm is to pick a *random* value for each variable, substitute the values into each expression, and evaluate. If the polynomials are equal, then the evaluations will necessarily coincide, but if the polynomials are not equal, then the

evaluations will likely disagree. By repeating the test a few times, we will either find out that the polynomials definitely differ, or else have very high confidence that the polynomials are equal.

A major question in complexity theory is whether there is a fast deterministic algorithm for polynomial identity testing. The standard approach to doing so is to derandomize the above randomized algorithm: Instead of evaluating the given polynomials with a random input, evaluate them with a pseudorandom input.

In the algebraic model, pseudorandom inputs take the following form: Instead of substituting each variable by a scalar, we substitute each variable by another polynomial. The polynomials in the substitution, collectively called a generator, crucially use many fewer variables than the original polynomials that we were given and have low degree. Because the number of variables is so much smaller and the degree is low, it becomes efficient again to check for equality by just expanding each of the resulting polynomials as a sum of monomials. For example, we might substitute

$$x \leftarrow (u-4)(u-3) \quad y \leftarrow (u-5)(u-3) \quad z \leftarrow (u-5)(u-4) \tag{1.4}$$

into (1.1) and (1.2) in which case they expand to $-4u^4+68u^3-428u^2+1180u-1200$. Plugging (1.4) into $xy+xz+yz$, meanwhile, yields $3u^4-48u^3+285u^2-744u+720$. Since the two results differ, we conclude that (1.1) and (1.2) are distinct from $xy+xz+yz$.

For any nontrivial generator, there exists a pair of polynomials that are distinct, but plugging the generator into them yields the same result. For example, the polynomial $-4yz$ is distinct from (1.1) and (1.2), but plugging (1.4) into it nevertheless produces the same $-4u^4+68u^3-428u^2+1180u-1200$. To derandomize polynomial identity testing, however, we can adapt the generator based on the polynomials we are given. The quest, therefore, is to build a sufficiently diverse array of generators and to recognize when one of those generators will work for the polynomials that are given.

Contributions In Chapter 2, we introduce a systematic approach to understanding the power of a generator, namely studying its *vanishing ideal*, the set of polynomials into which substituting the generator yields zero. For example, subtracting $-4yz$ from (1.1) and (1.2)

$$[xy - 2xz - 3yz] - [-4yz] = xy - 2xz + yz \tag{1.5}$$

yields a polynomial in the vanishing ideal of (1.4). Through the vanishing ideal we equate the derandomization of polynomial identity testing with lower bounds for algebraic computation.

As another conceptual contribution, we demonstrate the utility of generators that are rational functions rather than merely polynomials. Specifically, we introduce a generator based on the evaluation of rational functions. We show that this generator is equivalent to the commonly-used generator of Shpilka and Volkovich, and therefore inherits all its derandomization results.

We give two characterizations of the vanishing ideal for our generator, and through them, we capitalize on both directions of the derandomization–lower bounds equivalence. The first characterization is a small and explicit set of polynomials that jointly produce the vanishing ideal. As corollaries, we derive tight bounds for several parameters that are of interest in the context of derandomization. The second characterization, inspired by an alternating algebra representation, is a membership test for the vanishing ideal. Through this characterization, we rederive known derandomization results based on the generator of Shpilka and Volkovich. As a proof of concept of the vanishing ideal approach, we moreover prove a new derandomization result for a class of polynomials at the frontier of polynomial identity testing research.

Acknowledgement of Contributions The results in Chapter 2 will appear at ITCS 2022 [vMM22], and represent joint work with Dieter van Melkebeek.

1.2 Circuit Minimization

In Chapter 3, we prove lower bounds related to the problem of *circuit minimization*. Circuits model computation as a sequence of gates. The first few gates are set according to some input bits, and every subsequent gate computes the logical AND, OR, or NOT of preceding gates. Every circuit determines a function from the input gates to the output gates. In circuit minimization, we are given a target function as an explicit list of input-output mappings and must find a circuit of minimum size that computes it. More generally, there are a number of variations on circuits, and each one determines a different variation on circuit minimization.

Circuit minimization has received much attention, in part due to the role of circuits as a model for universal computation. Moreover, circuit minimization resists classification by one of the most successful theories in computational complexity, namely the theory of NP-completeness. Informally, a problem is an NP problem if it has solutions that can be verified efficiently. For example, while solving a given Sudoku board is challenging, verifying the validity of a completed board is easy. Similarly, circuit minimization is an NP problem, because it is easy to check whether a given circuit realizes each given input-output pair. In addition to logic puzzles and minimizing circuits, there are thousands of naturally occurring NP problems. We say that an NP problem is NP-complete if solving that problem would imply solving *every* NP problem. Among the naturally-occurring NP problems, the overwhelming majority are known to either have efficient algorithms, or else are NP-complete. Circuit minimization is among the few exceptions to this classification. We call such exceptions NP-intermediate problems.

Contributions In Chapter 3, we show that a particular variant of circuit minimization is at least as hard as another class of NP-intermediate problems, namely *isomorphism problems*. These include the most widely-recognized isomorphism problem, graph isomorphism, but also more difficult isomorphism problems, including some for which the best known algorithms require exponential time. The complexity of each of these isomorphism problems is

a lower bound for the complexity of circuit minimization. This presents a rare link between NP-intermediate problems.

Central to the improvement is a system for *verifying nonisomorphism*. We do so by exhibiting high entropy in a particular probability distribution, in a manner inspired by the theory of interactive proofs. We show how to validate the occurrence of this higher entropy assuming an oracle for the particular variant of circuit minimization.

Along the way, we show how to efficiently encode a sequence of independent samples from any probability distribution that is uniform on its support. The efficiency is best possible up to an error term that becomes insignificant as the number of samples increases.

An efficient randomized reduction was already known, but it had a nonzero probability of error. Our reduction has no such possibility of error. The improvement is interesting because traditional approaches to eliminating the error can only be accomplished for problems whose complements are also NP problems. Our approach circumvents this limitation.

Acknowledgement of Contributions The results in Chapter 3 appeared in ITCS 2018 [AGvM+18a] and in the SIAM Journal on Computing [AGvM+18b], and represent joint work with Eric Allender, Joshua Grochow, Dieter van Melkebeek, and Cristopher Moore.

1.3 Inversion Minimization

In the final chapter, we present lower bounds for *inversion minimization* in the comparison model. In this model, there is a set of items that are ranked in an unknown way, and problems amount to the discovery of some information about the ranking. Information is discovered by making comparison queries. These consist of selecting two distinct items, and asking which one has the higher rank. We are interested in the number of queries that are necessary to solve a given problem.

The comparison model is the standard model of computation for analyzing bread-and-butter algorithms like sorting, selection, and heap construction. Widely known is the text-

book result that sorting requires $\Omega(n \log n)$ comparisons, as well as that selection and heap construction can be done with $O(n)$ comparisons.

Another problem is *minimizing inversions in a tree*. In this problem, the items are arranged as the leaves of a rooted tree, and we want to order the tree so that the number of inversions between the ranks of the items and their order in the tree is minimized. We devised this problem for the undergraduate algorithms course at UW–Madison, and have since discovered that it appeared in the psychology literature forty years ago. Finding the minimizing order can be as hard as sorting, but finding the minimum number of inversions does not seem to have been analyzed.

Contributions We show that computing the minimum number of inversions in a tree is, in many cases, nearly as hard as sorting. In contrast to the well-known lower bounds in the comparison model, these lower bounds do not follow from information theoretic arguments, nor from a literal equivalence to sorting. Indeed, the complexity of the problem depends on the shape of the tree. A critical consideration is whether there is a large subtree whose leaves can be arranged in any order.

We introduce a lower bound framework based on a form of certificate complexity suited to the comparison model. We use connectivity considerations to develop some intuition about the nature of computation in the comparison model. We then import the notion of sensitivity from the Boolean query model to the comparison query model and show that sensitive functions require many queries to be computed.

Our lower bounds for minimizing inversions in a tree follow from lower bounds on the problem's sensitivity. We prove an unconditional lower bound that is tight to within a factor of three, and we improve it to one that is tight up to a small additive term, based on a combinatorial conjecture.

Acknowledgement of Contributions The results in Chapter 4 represent unpublished, joint work with Dieter van Melkebeek.

Chapter 2

Polynomial Identity Testing

2.1 Overview

Polynomial identity testing (PIT) is the fundamental problem of deciding whether a given multi-variate arithmetic circuit formally computes the zero polynomial. PIT has a simple efficient randomized algorithm that only needs black-box access to the circuit: Pick a random point and check whether the circuit evaluates to zero on that particular point.

In spite of the fundamental nature of PIT and the simplicity of the randomized algorithm, no efficient deterministic algorithm is known—even in the white-box setting, where the algorithm has access to the description of the circuit. The existence of such an algorithm would imply long-sought circuit lower bounds [HS80; Agr05; KI04]. Conversely, sufficiently strong circuit lower bounds yield blackbox derandomization for all of BPP, the class of decision problems admitting efficient randomized algorithms with bounded error [NW94; IW97]. Although the known results leave gaps between the two directions, they suggest that PIT acts as a BPP-complete problem in the context of derandomization, and that derandomization of BPP can be achieved in a blackbox fashion if at all.

Blackbox derandomization of PIT for a class of polynomials $\mathcal{C}$ in the variables $x_1, \ldots, x_n$ is equivalent to the efficient construction of a substitution G that replaces each x_i by a

low-degree polynomial in a small, common set of fresh variables such that, for every nonzero polynomial p from $\mathcal{C}$, $p(G)$ remains nonzero [SY10, Lemma 4.1]. We refer to G as a generator and the fresh variables as the seed, and we say that G hits the class $\mathcal{C}$. If p and G have degree at most $n^{O(1)}$, the resulting deterministic PIT algorithm for $\mathcal{C}$ makes $n^{O(l)}$ black-box queries.

Much progress on derandomizing PIT has been obtained by designing such substitutions and analyzing their hitting properties for interesting classes $\mathcal{C}$. Shpilka and Volkovich [SV15] introduced a generator, by now dubbed the Shpilka–Volkovich generator or "SV-generator" for short, and proved that it hits sums of a bounded number of read-once formulas for $l = O(\log n)$, later improved to $l = O(1)$ [MV18]. The generator for $l = O(\log n)$ has also been shown to hit multi-linear depth-4 circuits with bounded top fan-in [KMS+13], multi-linear bounded-read formulas [AvMV15], commutative read-once oblivious arithmetic branching programs [FSS14], $\Sigma\mathrm{m} \bigwedge \Sigma\Pi^{O(1)}$ formulas [For15], circuits with locally-low algebraic rank in the sense of [KS17], and orbits of simple polynomial classes under invertible linear transformations of the variables [MS21]. The generator is also an ingredient in other hitting set constructions, notably constructions using the technique of low-support rank concentration [ASS13; AGK+15; GKS+17; GKS17; ST21; BG21]. It also forms the core of a "succinct" generator that hits a variety of classes including depth-2 circuits [FSV17].

Vanishing Ideal In this chapter a systematic study of the power of the SV-generator is initiated. For any generator G, G hits a polynomial p if and only if the substitution $p(G)$ is nonzero. The power of G, therefore, is determined by the set of p such that $p(G)$ vanishes. This set, denoted $\mathrm{Van}[G]$, has the algebraic structure of an ideal, and is known as the *vanishing ideal* of G. The results in this chapter can be understood as precisely characterizing the vanishing ideal of the SV-generator for all choices of parameters.

There are two natural ways in which to apply a characterization of the vanishing ideal:

Derandomization To show that a generator G hits a class $\mathcal{C}$ of polynomials, it suffices (and

is necessary) to prove that the intersection of $\mathcal{C}$ with $\mathrm{Van}[G]$ consists of at most the zero polynomial. The vanishing ideal of SV tells us what polynomials we need to focus on when designing other generators for derandomizing PIT in combination with SV.

Lower bounds If we know that the generator G hits a class $\mathcal{C}$ of polynomials, any expression for a nonzero element of $\mathrm{Van}[G]$ yields an explicit polynomial that falls outside $\mathcal{C}$. Such a statement is often referred to as hardness of representation, and can be viewed as a lower bound in the model of computation underlying $\mathcal{C}$ (provided the polynomial can be computed in the model at all).

We will see both uses of this chapter's characterizations of the SV-generator.

Rational Function Evaluations Another contribution of this chapter is the development of an alternate view of the SV-generator, namely as evaluations of univariate rational functions of low degree. In addition to its intrinsic appeal and its applicability, the perspective facilitates the study of the vanishing ideal.

The transition goes as follows. The SV-generator takes as additional parameters a positive integer l, and a choice of distinct field elements a_i for each of the original variables x_i, $i \in [n]$. We refer to the elements a_i as *abscissas*, and denote the generator for a given value of l by SV^l (suppressing the choice of abscissas). SV^1 uses two fresh variables, y and z, and can be described succinctly in terms of the Lagrange interpolants L_i for the set of abscissas:

$$x_i \leftarrow z \cdot L_i(y) \doteq z \cdot \prod_{i' \in [n] \setminus \{i\}} \frac{y - a_{i'}}{a_i - a_{i'}}. \tag{2.1}$$

By rescaling, the denominators on the right-hand side of (2.1) can be cleared, resulting in the following somewhat simpler substitution:

$$x_i \leftarrow z \cdot \prod_{i' \in [n] \setminus \{i\}} (y - a_{i'}). \tag{2.2}$$

The vanishing ideals of (2.2) and SV^1 are the same up to rescaling the variables to match the rescaling from (2.1) to (2.2).

More importantly, we apply the change of variables $z \leftarrow z' / \prod_{i \in [n]} (y - a_i)$, resulting in a substitution that now uses rational functions:

$$x_i \leftarrow \frac{z'}{y - a_i}. \tag{2.3}$$

The notion of vanishing ideal naturally extends to rational function substitutions. The change of variables from (2.2) to (2.3) establishes that any polynomial vanishing on (2.2) also vanishes on (2.3). The change of variables is invertible (the inverse is $z' \leftarrow z \prod_{i \in [n]} (y - a_i)$), so any polynomial vanishing on (2.3) also vanishes on (2.2). Therefore the vanishing ideal of (2.3) is the same as that of SV^1 up to rescaling the variables.

Note that, for fixed y and z', (2.3) may be interpreted as first forming a univariate rational function $f(\alpha) = \frac{z'}{y - \alpha}$ (depending on y and z', but independent of i) and then substituting $x_i \leftarrow f(a_i)$. As y and z' vary, f ranges over all rational functions with numerator degree zero and denominator degree one. We thus denote (2.3) by RFE_1^0, where RFE is a shorthand for *Rational Function Evaluation*, 0 bounds the numerator degree, and 1 bounds the denominator degree.

As a generator, RFE_1^0 naturally generalizes to RFE_l^k for arbitrary $k, l \in \mathbb{N}$:

Definition 2.1 (RFE Generator). Let $\mathbb{F}$ be a field and $X \doteq \{x_i : i \in [n]\}$ a set of variables. The *Rational Function Evaluation (RFE) Generator* for $\mathbb{F}[X]$ is parametrized by the following data:

- For each $i \in [n]$, a distinct *abscissa* $a_i \in \mathbb{F}$.
- A non-negative integer k, the *numerator degree*.
- A non-negative integer l, the *denominator degree*.

The generator takes as seeds rational functions $f \in \mathbb{F}(\alpha)$ such that f can be written g/h for some $g, h \in \mathbb{F}[\alpha]$ with $\deg(g) \leq k$, $\deg(h) \leq l$, and $h(a_i) \neq 0$ for all $i \in [n]$. From a seed f, it generates the substitution $x_i \leftarrow f(a_i)$ for each $i \in [n]$. ◂

There are multiple ways to parametrize the seed of RFE_l^k using scalars, such as by specifying coefficients, evaluations, or roots for each of the numerator and denominator. The flexibility to choose is a source of convenience. Section 2.2 discusses different parametrizations, as well as how to obtain deterministic black-box PIT algorithms from the generator, and the required size of the underlying field $\mathbb{F}$. As is customary in the context of black-box derandomization of PIT, we will assume that $\mathbb{F}$ is sufficiently large, possibly by taking a field extension.

The connection between RFE_1^0 and SV^1 extends as follows. For higher values of l, SV^l is defined as the sum of l independent instantiations of SV^1. The same transformations as above relate SV^l and the sum of l independent instantiations of RFE_1^0. The latter in turn is equivalent to RFE_l^{l-1} by partial fraction decomposition. The conclusion is that SV^l is equivalent, up to variable rescaling, to RFE_l^{l-1}. Section 2.2 presents a formal treatment.

For parameter values $k \neq l-1$, there is no SV-generator that corresponds to RFE_l^k, but $\mathrm{SV}^{\max(k+1,l)}$ encompasses RFE_l^k (up to rescaling) and uses at most twice as long of a seed. Thus, the RFE-generator and the SV-generator efficiently hit the same classes of polynomials. However, RFE's simpler univariate dependence on the abscissas—as opposed to SV's multi-variate dependence—enables our approach for determining the vanishing ideal. The moral is that, even though polynomial substitutions are sufficient for derandomizing PIT, it nevertheless helps to consider rational substitutions. Their use may simplify analysis, and arguably yield more elegant constructions.

Generating Set The first characterization of the vanishing ideal of RFE is a small and explicit generating set for the vanishing ideal of RFE. It consists of instantiations of a single determinant expression.

Theorem 2.2. *Let $k, l \in \mathbb{N}$, $\{x_i : i \in [n]\}$ be a set of variables, and a_i for $i \in [n]$ be distinct field elements. The vanishing ideal of* RFE_l^k *over the given set of variables for the given choice of abscissas $(a_i)_{i\in[n]}$ is generated by the following polynomials over all choices of*

$k+l+2$ variable indices $i_1, i_2, \ldots, i_{k+l+2} \in [n]$:

$$\mathrm{EVC}_l^k[i_1, i_2, \ldots, i_{k+l+2}] \doteq \det\begin{bmatrix} a_{i_j}^l x_{i_j} & a_{i_j}^{l-1} x_{i_j} & \ldots & x_{i_j} & a_{i_j}^k & a_{i_j}^{k-1} & \ldots & 1 \end{bmatrix}_{j=1}^{k+l+2}. \tag{2.4}$$

Moreover, the polynomials $\mathrm{EVC}_l^k[i_1, i_2, \ldots, i_{k+l+2}]$ form a generating set of minimum size when $\{i_1, i_2, \ldots, i_{k+l+2}\}$ ranges over all subsets of $[n]$ of size $k+l+2$ that contain a fixed set $C \subseteq [n]$ of $k+1$ variable indices, and $i_1 < i_2 < \cdots < i_{k+l+2}$.

The name "EVC" is a shorthand for "Elementary Vandermonde Circulation". Later we shall see a representation of polynomials using alternating algebra, which connects with notions from network flow. In this representation, polynomials in the vanishing ideal coincide with circulations, and instantiations of EVC are the elementary circulations.

We call the set C in Theorem 2.2 a *core*. The core C plays a similar role as in a sunflower except that, unlike the petals of a sunflower, the various sets S do not need to be disjoint outside the core.

Example 2.3. Consider the special case where $k = 0$ and $l = 1$. The generator for RFE_1^0 when $i_1 = 1$, $i_2 = 2$, and $i_3 = 3$ is given by

$$\mathrm{EVC}_1^0[1,2,3] \doteq \begin{vmatrix} a_1 x_1 & x_1 & 1 \\ a_2 x_2 & x_2 & 1 \\ a_3 x_3 & x_3 & 1 \end{vmatrix} = (a_1 - a_2) x_1 x_2 + (a_2 - a_3) x_2 x_3 + (a_3 - a_1) x_3 x_1. \tag{2.5}$$

For any fixed $i^* \in [n]$, the polynomials $\mathrm{EVC}_1^0[i_1, i_2, i_3]$ form a generating set of minimum size when $\{i_1, i_2, i_3\}$ ranges over all subsets of $[n]$ that contain $C = \{i^*\}$, and $i_1 < i_2 < i_3$. ◂

In general, the generators EVC_l^k are nonzero multi-linear homogeneous polynomials of degree $l+1$ containing all multi-linear monomials of degree $l+1$.

Each generating set of minimum size in Theorem 2.2 yields a Gröbner basis with respect to every monomial order that prioritizes the variables outside C. A Gröbner basis is a special basis that allows solving ideal-membership queries more efficiently as well as solving systems of polynomial equations [CLO13]. Computing Gröbner bases for general ideals

is exponential-space complete. Theorem 2.2 represents a rare instance of a natural and interesting ideal for which we know an explicit Gröbner basis.

To gain some intuition about dependencies between the generators EVC_l^k, note that permuting the order of the variables used in the construction of EVC_l^k yields the same polynomial or minus that polynomial, depending on the sign of the permutation. This follows from the determinant structure of EVC_l^k, and is the reason why we need to fix the order of the variables in order to obtain a generating set of minimum size. More profoundly, the following relationship holds for every choice of $k+l+3$ indices $i_1, i_2, \ldots, i_{k+l+3} \in [n]$ and every univariate polynomial w of degree at most k:

$$\det\Big[w(a_{i_j}) \quad a_{i_j}^l x_{i_j} \quad a_{i_j}^{l-1} x_{i_j} \quad \ldots \quad x_{i_j} \quad a_{i_j}^k \quad a_{i_j}^{k-1} \quad \ldots \quad 1\Big]_{j=1}^{k+l+3} = 0. \tag{2.6}$$

The determinant in (2.6) vanishes because the first column of the matrix is a linear combination of the last $k+1$. A Laplace expansion across the first column allows us to write the determinant of the matrix as a linear combination of minors, and each minor is an instantiation of EVC_l^k. As the determinant vanishes, (2.6) represents a linear dependency for every nonzero polynomial w of degree at most k. In fact, when $\{i_1, \ldots, i_{k+l+3}\}$ varies over subsets of $[n]$ containing a fixed core of size $k+1$, the equations (2.6) generate all linear dependencies among instances of EVC_l^k.

As corollaries to Theorem 2.2 we obtain the following tight bounds on $\mathrm{Van}[\mathrm{RFE}_l^k]$. The bounds hold for all choices of parameters, as long as the abscissas for different variables remain distinct.

- The minimum *degree* of a nonzero polynomial in $\mathrm{Van}[\mathrm{RFE}_l^k]$ equals $l+1$. This proves a conjecture by Fournier and Korwar [FK18] (additional partial results reported in [Kor21]) that there exists a polynomial of degree $l+1$ in $n = 2l+1$ variables that SV^l fails to hit. The conjecture follows because the generators for $\mathrm{Van}[\mathrm{SV}^l]$ have degree $l+1$ and use $2l+1$ variables.

As none of the generators contain a monomial of support l or less, the same holds for every nonzero polynomial in $\mathrm{Van}[\mathrm{RFE}_l^k]$. This extends the known property that SV^l hits every polynomial that contains a monomial of support l or less [SV15].

- The minimum *sparseness*, i.e., number of monomials, of a nonzero polynomial in $\mathrm{Van}[\mathrm{RFE}_l^k]$ equals $\binom{k+l+2}{l+1}$. The generators EVC_l^k realize the bound as they exactly contain all multi-linear monomials of degree $l+1$ that can be formed out of their $k+l+2$ variables.

  The claim that no nonzero polynomial in $\mathrm{Van}[\mathrm{RFE}_l^k]$ contains fewer than $\binom{k+l+2}{l+1}$ monomials requires an additional combinatorial argument. It is a (tight) quantitative strengthening of the well-known property that SV^l hits every polynomial with fewer than 2^l monomials [AvMV15; GKS+17; For15; FSV17]. Note that for $k = l-1$ we have that $\binom{k+l+2}{l+1} = \binom{2l+1}{l+1} = \Theta(2^{2l}/\sqrt{l})$.

- The minimum *partition class size* of a nonzero set-multi-linear polynomial of degree $l+1$ in $\mathrm{Van}[\mathrm{RFE}_l^k]$ equals $k+2$. Set-multi-linearity is a common restriction in works on derandomizing PIT and arithmetic circuit lower bounds. A polynomial p of degree $l+1$ in a set of variables $\{x_1, \ldots, x_n\}$ is said to be set-multi-linear if $[n]$ can be partitioned as $[n] = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_{l+1}$ such that every monomial in p is a product $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_{l+1}}$, where $i_j \in X_j$. Note that set-multi-linearity implies multi-linearity but not the other way around.

  As the generators EVC_l^k are not set-multi-linear, it is not immediately clear from Theorem 2.2 whether $\mathrm{Van}[\mathrm{RFE}_l^k]$ contains nontrivial set-multi-linear polynomials. However, a variation on the construction of the generators EVC_l^k yields explicit set-multi-linear homogeneous polynomials in $\mathrm{Van}[\mathrm{RFE}_l^k]$ of degree $l+1$ where each X_j has size $k+2$. We denote them by ESMVC_l^k, where ESMVC stands for "Elementary Set-Multi-linear Vandermonde Circulation". ESMVC_l^k contains all monomials of the form $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_{l+1}}$ with $i_j \in X_j$. For any variable partition $X_1 \sqcup X_2 \sqcup \cdots \sqcup X_{l+1}$ with $|X_1| = \cdots = |X_{l+1}| = k+2$,

ESMVC_l^k is the only set-multi-linear polynomial $\mathrm{Van}[\mathrm{RFE}_l^k]$ with that variable partition, up to a scalar multiple.

Membership Test The second characterization of the vanishing ideal of RFE can be viewed as a structured membership test. There is a generic way to obtain a deterministic membership test for the vanishing ideal of any hitting set generator G, namely the well-known transformation of a hitting set generator into a deterministic blackbox PIT algorithm [Ore22; DL78; Zip79; Sch80]. When G is a polynomial substitution using l seed variables that hits the n-variate polynomials p in a class $\mathcal{C}$, the deterministic black-box PIT algorithm for $\mathcal{C}$ makes no more than $n^{O(l)}$ queries as long as p and G have degree $n^{O(1)}$. By clearing denominators, the same follows for rational substitutions like RFE_l^k, which can be parametrized to use only $k+l+1$ seed variables. The resulting deterministic algorithm decides PIT for $p \in \mathcal{C}$ provided RFE_l^k hits $\mathcal{C}$. Unconditionally, the algorithm decides membership of any p to the vanishing ideal $\mathrm{Van}[\mathrm{RFE}_l^k]$.

Capitalizing on the generating set of Theorem 2.2, we develop a more structured deterministic membership test for $\mathrm{Van}[\mathrm{RFE}_l^k]$. In the important case of multi-linear polynomials, the test takes the following form.

Theorem 2.4. *Let $k, l \in \mathbb{N}$, $\{x_i : i \in [n]\}$ be a set of variables, a_i for $i \in [n]$ be distinct field elements, and Z a set of at least $n-k-l-1$ nonzero field elements. A multi-linear polynomial p in those variables belongs to $\mathrm{Van}[\mathrm{RFE}_l^k]$ if and only if both of the following conditions hold:*

1. *p has no homogeneous components of degree l or less, nor of degree $n-k$ or more.*
2. *For all disjoint subsets $K, L \subseteq [n]$ with $|K| = k$ and $|L| = l$, and every $z \in Z$, $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ evaluates to zero upon the following substitution for each $i \in \overline{K \cup L}$*

$$x_i \leftarrow z \cdot \frac{\prod_{i' \in K}(a_i - a_{i'})}{\prod_{i' \in L}(a_i - a_{i'})}. \tag{2.7}$$

The first part of condition 1 in Theorem 2.4 extends the well-known property that SV^l hits every multi-linear polynomial that contains a monomial of degree l or less. Combined with the second part of the condition, it implies that all multi-linear polynomials on $n \leq k + l + 1$ variables are hit by RFE_l^k.

In condition 2, $\left(\frac{\partial p}{\partial L}\right)\big|_{K \leftarrow 0}$ denotes the polynomial obtained by taking the partial derivative of p with respect to every variable in L, and setting all the variables in K to zero. (The order of the operations does not matter, and the resulting polynomial depends only on variables in $\overline{K \cup L}$.)

Several prior papers demonstrated the utility of partial derivatives and zero substitutions in the context of derandomizing PIT using the SV-generator, especially for syntactically multi-linear models [SV15; KMS+13; AvMV15]. By judiciously choosing variables for those operations, these papers managed to simplify p and reduce PIT for p to PIT for simpler instances, resulting in an efficient recursive algorithm. In Section 2.4, a general framework for such algorithms is developed, and correctness is proved directly from Theorem 2.4. For every multi-linear polynomial p hit by RFE_l^k, the sets K and L in Theorem 2.4 describe how to choose k zero substitutions and l derivatives so that a recursive approach shows that p is hit by RFE_l^k. It follows that any argument that SV or RFE hit a class of multi-linear polynomials can, in principle, be converted into one based on zero substitutions and partial derivatives. Thus, Theorem 2.4 shows that these tools harness the complete power of SV and RFE for multi-linear polynomials.

Applications We illustrate the utility of our characterizations of the vanishing ideal of RFE in the two directions mentioned before.

Derandomization To start, we will see how Theorem 2.4 yields an alternate proof of the result from [MV18] that every nonzero read-once formula F is hit by SV^1, or equivalently, by RFE_1^0. Whereas the original proof hinges on a clever ad-hoc argument, the proof

in this chapter (described in Section 2.4) is entirely systematic and amounts to a couple straightforward observations in order to apply Theorem 2.4.

Next, as a proof of concept of the additional power of our characterization for derandomization, we develop an improvement in the model of read-once oblivious algebraic branching programs (ROABPs).

Theorem 2.5. *For every $l \in \mathbb{N}$, SV^l hits the class of polynomials computed by read-once oblivious algebraic branching programs of width less than $1 + (l/3)$ that contain a monomial of degree at most $l + 1$.*

To the best of our knowledge, Theorem 2.5 is incomparable to the known results for ROABPs [RS05; JQS09; JQS10; FS13; FSS14; AGK+15; AFS+18; GKS+17; GKS17; GG20; ST21; BG21]. Without the restriction that the polynomial has a monomial of degree at most $l + 1$, Theorem 2.5 would imply a fully blackbox polynomial-time identity test for the class of constant-width ROABPs. No such test has been proven to exist at this time; prior work requires either quasipolynomial time or requires opening the blackbox, such as by knowing the order in which the variables are read.

With the restriction, the known property that SV^{l+1} hits every polynomial containing a monomial of support $l + 1$ or less [SV15] implies that SV^{l+1} hits the class $\mathcal{C}$ in Theorem 2.5. The result can thus be viewed as an improvement from SV^{l+1} to SV^l. Even though the improvement is modest from this perspective, it is a substantial step toward a fully polynomial-time identity test for constant-width ROABPs. This is because the vanishing ideal of SV^l contains—and is generated by—polynomials that satisfy the restriction, namely instances of EVC_l^{l-1}. Theorem 2.5 establishes that none of the generators, nor any nonzero linear combination thereof, is computed by a small-width ROABP. If that can be extended to combinations of generators with coefficients that are arbitrary polynomials, then Theorem 2.5 would hold without the additional restriction, and a fully blackbox polynomial-time test for constant-width ROABPs would follow.

In addition, the method of proof of Theorem 2.5 diverges significantly from prior uses of the SV-generator. Most prior uses of the SV-generator rely on combinatorial arguments, i.e., arguments that depend only on which monomials are present in polynomials of $\mathcal{C}$. Theorem 2.5 necessarily goes beyond this, because there is a polynomial in $\mathrm{Van}[\mathrm{SV}^l]$ of degree $l+1$ that has the same monomials as a polynomial computed by an ROABP of width 2. Namely, any instance of ESMVC_l^{l-1} contains exactly all the monomials of the form $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_{l+1}}$ with $(i_1, \ldots, i_{l+1}) \in X_1 \times \cdots \times X_{l+1}$ for some disjoint sets X_j; the same goes for $\prod_j \sum_{i_j \in X_j} x_{i_j}$, which is computed by an ROABP of width 2.

Lower Bounds The argument in the previous paragraph also illustrates this direction: the derandomization result for the class $\mathcal{C}$ implies that every ROABP computing EVC_l^{l-1}, ESMVC_l^{l-1}, or any other polynomial of degree $l+1$ in the vanishing ideal, has width at least $1 + (l/3)$. Other hardness of representation results follow in a similar manner from prior hitting properties of SV in the literature. The following lower bounds apply to computing both EVC_l^{l-1} and ESMVC_l^{l-1}:

- Any syntactically multi-linear formula must have at least $\Omega(\log(l)/\log\log(l))$ reads of some variable [AvMV15, Theorem 6.3].
- Any sum of read-once formulas must have at least $\Omega(l)$ terms [MV18, Corollary 5.2].
- There exists an order of the variables such that any ROABP with that order must have width at least $2^{\Omega(l)}$ [FSS13, Corollary 4.3].
- Any $\Sigma\mathrm{m}\bigwedge\Sigma\Pi^{O(1)}$ formula must have top fan-in at least $2^{\Omega(l)}$ [For15]. See [FSV18, Lemma 5.12].
- Lower bounds in characteristic zero for circuits with locally-low algebraic rank [KS17, Lemma 5.2].

Techniques A recurring tool is the analysis of the coefficients of the Laurent expansion of $p(\mathrm{RFE}_l^k)$ around certain abscissas. The technique is captured in Lemma 2.17, which we call the Zoom Lemma. A proof from first principles is provided that requires no knowledge of Laurent expansions. The Zoom Lemma is used in the proofs of all of Theorems 2.2, 2.4, and 2.5, as well as several of the other results. The basic logic is to zoom in on the projection of p onto certain monomials on a subset of the variables, and show that if the projection does not vanish at a certain point, then a particular Laurent coefficient of $p(\mathrm{RFE}_l^k)$ is nonzero, and therefore RFE_l^k hits p.

Theorem 2.2 states the equality of two ideals: $\langle \mathrm{EVC}_l^k \rangle = \mathrm{Van}[\mathrm{RFE}_l^k]$, where $\langle \mathrm{EVC}_l^k \rangle$ denotes the ideal generated by all instantiations of EVC_l^k, and $\mathrm{Van}[\mathrm{RFE}_l^k]$ the vanishing ideal of RFE_l^k.

- The inclusion $\subseteq$ follows from linearizing the defining equations of RFE_l^k. This is where the univariate dependency on the abscissas comes into play.

- To establish the inclusion $\supseteq$ we first show that every equivalence class of polynomials modulo $\langle \mathrm{EVC}_l^k \rangle$ contains a representative p whose monomials exhibit the combinatorial structure of a core. The structure enables the Zoom Lemma to exhibit a particular Laurent coefficient of $p(\mathrm{RFE}_l^k)$ that receives a contribution from just a single monomial. As there are no other contributions that can cancel out that one contribution, the coefficient is nonzero, whence RFE_l^k hits p.

The proof of Theorem 2.4 also relies on Laurent expansions through the Zoom Lemma. Membership to the ideal is equivalent to the vanishing of all coefficients of the expansion. The proof can be viewed as determining a small number of coefficients sufficient to guarantee that their vanishing implies all coefficients vanish. The restriction to multi-linear polynomials p allows us to express the projections of p as the result of applying partial derivatives and zero-substitutions.

Theorem 2.5 makes use of the characterization of the minimum width of a read-once oblivious arithmetic branching program computing a polynomial p as the maximum rank of the monomial coefficient matrices of p for various variable partitions [Nis91]. We reduce to the case where p is homogeneous of degree $l+1$, whence the monomial coefficient matrices have a block-diagonal structure. An application of the Zoom Lemma in the contrapositive yields linear equations between elements of consecutive blocks under the assumption that SV^l fails to hit p. When some block is zero, the equations yield a Cauchy system of equations on the rows or columns of its neighboring blocks; since Cauchy systems have full rank, we deduce severe constraints on the row-space/column-space of the neighboring blocks. A careful analysis turns this observation into a rank lower bound of at least $1 + (l/3)$ for a well-chosen partition of the variables.

In this application the Zoom Lemma is instantiated several times in parallel to form a large system of equations on the coefficients of p, and the whole system is needed for the analysis. This stands in contrast to most prior work using SV, which uses knowledge of how p is computed to guide a search for a *single* fruitful instantiation of the Zoom Lemma.

Alternating Algebra Representation The inspiration for several of this chapter's results stems from expressing the polynomials EVC_l^k using concepts from alternating algebra (also known as exterior algebra or Grassmann algebra). In fact, Theorem 2.4 hinges on the relationship $\partial^2 = 0$ from alternating algebra. An earlier form of the statement and proof of the theorem made use of that framework, but we managed to eliminate the alternating algebra afterwards. Still, the perspective is insightful and potentially helpful for future developments, so the connection is presented here. We shall see the intuition behind Theorem 2.4 in the simple case where the degree of the polynomial p equals $l+1$. In that setting, belonging to the ideal generated by the polynomials EVC_l^k is equivalent to being in their linear span.

The alternating algebra A of a vector space V over a field $\mathbb{F}$ consists of the closure of V

under an additional binary operation, referred to as 'wedge' and denoted $\wedge$, which is bilinear, associative, and satisfies

$$v \wedge v = 0 \tag{2.8}$$

for every $v \in V$. This determines a well-defined algebra. When the characteristic of $\mathbb{F}$ is not 2, this can equivalently be understood as

$$v_1 \wedge v_2 = -(v_2 \wedge v_1) \tag{2.9}$$

for every $v_1, v_2 \in V$. In any case, for any $v_1, v_2, \ldots, v_k \in V$,

$$v_1 \wedge v_2 \wedge \cdots \wedge v_k \tag{2.10}$$

is nonzero iff the v_i's are linearly independent, and any permutation of the order of the vectors in (2.10) yields the same element of A up to a sign. The sign equals the sign of the permutation, whence comes the name "alternating algebra." If V has a basis X of size n, then a basis for A can be formed by all 2^n expressions of the form (2.10) where the v_i's range over all subsets of X, and are taken in some fixed order. Considering the elements of X as vertices, the basis elements of A can be thought of as the oriented simplices of all dimensions that can be built from X.

Anti-commutativity, the relation (2.9), arises naturally in the context of network flow, where X denotes the vertices of the underlying graph, and a wedge $v_1 \wedge v_2$ of level $k = 2$ represents one unit of flow from v_1 to v_2. Equation (2.9) reflects the fact that one more unit of flow from v_1 to v_2 is equivalent to one less unit of flow from v_2 to v_1. The adjacent levels $k = 1$ and $k = 3$ also have natural interpretations in the flow setting: v_1 (the element of A of the form (2.10) with $k = 1$) represents one unit of surplus flow at v_1 (the vertex of the graph), and $v_1 \wedge v_2 \wedge v_3$ abstracts an elementary circulation of one unit along the directed cycle $v_1 \to v_2 \to v_3 \to v_1$.

The different levels are related by so-called boundary maps. Boundary maps are linear transformations that map a simplex to a linear combination of its subsimplices of one

dimension less. The maps are parametrized by a weight function $w : X \to \mathbb{F}$, and defined by

$$\partial_w : v_1 \wedge v_2 \wedge \cdots \wedge v_m \mapsto \sum_{i=1}^{m} (-1)^{i+1} w(v_i)\, v_1 \wedge \cdots \wedge v_{i-1} \wedge v_{i+1} \wedge \cdots \wedge v_m, \tag{2.11}$$

an expression resembling the Laplace expansion of a determinant along a column $[w(v_i)]_{i=1}^{m}$. In the flow setting, using $w \equiv 1$, $\partial_1(v_1 \wedge v_2 \wedge v_3)$ is the superposition of the three edge flows that make up one unit of circulation along the directed cycle $v_1 \to v_2 \to v_3 \to v_1$, and $\partial_1(v_1 \wedge v_2)$ is the superposition of surplus at v_1 and demand at v_2 corresponding to one unit of flow from v_1 to v_2. A linear combination p of terms (2.10) with $k = 2$ represents a valid circulation iff it satisfies conservation of flow at every vertex, which can be expressed as $\partial_1(p) = 0$, i.e., p is in the kernel of ∂_1. An equivalent criterion is for p to be the superposition of circulations along 3-cycles, which can be expressed as p being in the image of ∂_1. The relationship between the image and the kernel of boundary maps holds in general:

$$\operatorname{Im}\left(\partial_{w_m} \circ \partial_{w_{m-1}} \circ \cdots \circ \partial_{w_0}\right) \;=\; \bigcap_{i=0}^{m} \operatorname{Ker}\left(\partial_{w_i}\right). \tag{2.12}$$

In the context of the generators EVC_l^k, the set X creates a vertex for each variable, and simplices correspond to multi-linear monomials. The anti-commutativity of $\wedge$ coincides with the fact that swapping two arguments means swapping two rows in (2.4), which changes the sign of the determinant. Using the above boundary maps, the right-hand side of (2.4) can be viewed as $\partial_\omega(v_{i_1} \wedge v_{i_2} \wedge \cdots \wedge v_{i_{k+l+2}})$, where $\partial_\omega \doteq \partial_{w_k} \circ \partial_{w_{k-1}} \circ \cdots \circ \partial_{w_0}$ and $w_d(v_i) \doteq (a_i)^d$. By (2.12), this means that EVC_l^k is in the kernel of ∂_{w_d} for each $d \in \{0, 1, \ldots, k\}$, or equivalently, in the kernel of $\partial_{\widetilde{w}}$ for each $\widetilde{w} : X \to \mathbb{F}$ of the form $\widetilde{w}(v_i) = w(a_i)$ where w is a polynomial of degree at most k. This is precisely the condition (2.6). In fact, (2.12) implies that the linear span of the generators EVC_l^k consists exactly of the polynomials of degree $l+1$ in this kernel. The latter condition is precisely what the criterion in Theorem 2.4 expresses.

Organization Section 2.2 discusses details related to the use of RFE as a hitting set generator and presents a formal treatment of the relationship between RFE and SV. The generating set for the vanishing ideal (Theorem 2.2) is developed in Section 2.3, and the

ideal membership test (Theorem 2.4) is developed in Section 2.4. The results on sparseness are presented in Section 2.5, and the ones on set-multi-linearity are presented in Section 2.6. Background on ROABPs and the result on derandomizing PIT for ROABPs (Theorem 2.5) are covered in Section 2.7. We end this chapter in Section 2.8 with a further discussion of the alternating algebra representation.

2.2 Rational Function Evaluation Generator

We introduced our generator RFE in Definition 2.1 in the overview. In this section we discuss some details related to the use of RFE as a hitting set generator and its relationship with SV.

RFE as a Hitting Set Generator An ambiguity in Definition 2.1 is how to parametrize the seed by scalars. There are qualitatively distinct ways to go about this. They are all equivalent over large enough fields, however, so the option to choose is a source of convenience. Some natural parametrizations are the following:

Coefficients. Select scalars $g_0, \ldots, g_k, h_0, \ldots, h_l \in \mathbb{F}$ and set

$$f(\alpha) = \frac{g_k\alpha^k + g_{k-1}\alpha^{k-1} + \cdots + g_1\alpha + g_0}{h_l\alpha^l + h_{l-1}\alpha^{l-1} + \cdots + h_1\alpha + h_0}, \tag{2.13}$$

ignoring choices of $h_0, \ldots, h_l$ for which the denominator vanishes on some a_x.

Evaluations. Fix two collections, $B = \{b_1, \ldots, b_{k+1}\}$ and $C = \{c_1, \ldots, c_{l+1}\}$, each of distinct scalars from $\mathbb{F}$. Then select scalars $g_1, \ldots, g_{k+1}$ and $h_1, \ldots, h_{l+1}$ and set

$$f(\alpha) = \frac{g(\alpha)}{h(\alpha)} \tag{2.14}$$

where g is the unique degree-k polynomial with $g(b_1) = g_1$, $g(b_2) = g_2$, $\ldots$, $g(b_{k+1}) = g_{k+1}$, and h is defined similarly with respect to C. Choices of $h_1, \ldots, h_{l+1}$ that imply $h(a_i) = 0$ for some $i \in [n]$ are ignored.

Note that an explicit formula for g and h in terms of the parameters can be obtained using the Lagrange interpolants with respect to B and C.

Roots. Select scalars $z, s_1, \ldots, s_{k'}, t_1, \ldots, t_{l'} \in \mathbb{F}$ for some $k' \leq k$ and $l' \leq l$ and set

$$f(\alpha) = z \cdot \frac{(\alpha - s_1) \cdot \cdots \cdot (\alpha - s_{k'})}{(\alpha - t_1) \cdot \cdots \cdot (\alpha - t_{l'})}, \tag{2.15}$$

where $\{t_1, \ldots, t_{l'}\}$ is disjoint from $\{a_i : i \in [n]\}$.

In fact, it is no loss of power to restrict to $k' = k$ and $l' = l$.

Hybrids are of course possible, too. For example, the equivalence between RFE and SV (formalized in Lemma 2.8 in the next section) uses the evaluations parametrization for the numerator and roots parametrization for the denominator.

Quantitative bounds on the number of substitutions to perform follow from the following extension of the corresponding well-known result for polynomials [Ore22; DL78; Zip79; Sch80]:

Lemma 2.6. *Let $\mathbb{F}$ be field, and $f = g/h \in \mathbb{F}(\tau_1, \ldots, \tau_l)$ be a rational function in l variables with $\deg(g) \leq d$ and $\deg(h) \leq d$. Let $S \subseteq \mathbb{F}$ be finite. Then the probability that f vanishes or is undefined when each τ_i is substituted by a uniformly random element of S is at most $2d/|S|$.*

If $\mathbb{F}$ is not large enough to allow making the probability bound in Lemma 2.6 sufficiently small, we work with a sufficiently large extension field of $\mathbb{F}$ instead of $\mathbb{F}$ itself.

In this chapter, we analyze RFE by using fresh formal variables in the above parametrizations of the seed, and calculate in the field of rational functions in those variables. Evaluating $p(\mathrm{RFE}_l^k)$ for a polynomial p thus yields a rational function in those variables. Lemma 2.6 justifies that as long as $\mathbb{F}$ is large enough, this rational function is the zero rational function if and only if for every choice of scalars for the seed parameters, $p(\mathrm{RFE}_l^k)$ is zero.

Equivalence between RFE and SV The Shpilka-Volkovich generator can be defined as follows in the format of our definition of RFE.

Definition 2.7 (SV Generator). The *Shpilka–Volkovich (SV) Generator* for polynomials in the variables $x_1, \ldots, x_n$ is parametrized by the following data:

- For each $i \in [n]$, a distinct $a_i \in \mathbb{F}$.
- A positive integer, l.

The generator takes as seed l pairs of scalars $(y_1, z_1), \ldots, (y_l, z_l)$ and substitutes

$$x_i \leftarrow \sum_{j=1}^{l} \left(z_j \cdot \prod_{i' \in [n] \setminus \{i\}} \frac{y_j - a_{i'}}{a_i - a_{i'}} \right). \tag{2.16}$$

◂

We abbreviate the generator to SV^l or just SV.

Shpilka and Volkovich designed SV^l so that any selection of l of the variables could remain independent while the others were forced to zero. This can be viewed as an algebraic version of l-wise independence. SV^1 was realized with two seed variables, y and z, using Lagrange interpolation. The fresh variable y enables selecting one of the original variables x_i, namely by setting $y = a_i$. The selected variable x_i is then set to z, while the other variables are set to zero. For larger l, SV^l is the sum of l independent copies of SV^1.

As we sketched in Section 2.1, there is a close relationship between SV^l and RFE_l^{l-1}.

Lemma 2.8. *Let $\{x_1, \ldots, x_n\}$ be a set of variables and $l \geq 1$. There is an invertible diagonal transformation $A : \mathbb{F}^n \to \mathbb{F}^n$ such that, for any polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$, $p(\mathrm{SV}^l) = 0$ if and only if $(p \circ A)(\mathrm{RFE}_l^{l-1}) = 0$.*

In particular, the vanishing ideals of RFE_l^{l-1} and of SV^l are the same up to the rescaling of Lemma 2.8.

Proof of Lemma 2.8. Let $\widehat{\mathbb{F}}$ be the field of rational functions in indeterminates $v_1, \ldots, v_l$, $\zeta_1, \ldots, \zeta_l$ over $\mathbb{F}$. A polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ has $p(\mathrm{SV}^l) = 0$ if and only if p vanishes at the point

$$\left(\sum_{j=1}^{l} \zeta_j \prod_{i' \in [n] \setminus \{i\}} \frac{v_j - a_{i'}}{a_i - a_{i'}} \ : \ i \in [n] \right) \in \widehat{\mathbb{F}}^n. \tag{2.17}$$

Set $A : \mathbb{F}^n \to \mathbb{F}^n$ to be the diagonal linear transformation that divides the coordinate for x_i by $\prod_{i' \in [n] \setminus \{i\}} (a_i - a_{i'})$. It is invertible. Applying A^{-1} to (2.17) yields the point

$$\left(\sum_{j=1}^{l} \zeta_j \prod_{i' \in [n] \setminus \{i\}} (v_j - a_{i'}) \ : \ i \in [n] \right) = \left(\sum_{j=1}^{l} \left(\zeta_j \prod_{i' \in [n]} (v_j - a_{i'}) \right) \frac{1}{v_j - a_i} \ : \ i \in [n] \right). \tag{2.18}$$

p vanishes at (2.17) if and only if $p \circ A$ vanishes at (2.18). Now let $\widehat{\mathbb{F}}'$ be the field of rational functions in indeterminates $\tau_1, \ldots, \tau_l, \sigma_1, \ldots, \sigma_l$ over $\mathbb{F}$. After the invertible change of variables

$$\zeta_j \leftarrow \frac{1}{\prod_{i' \in [n]} (\tau_j - a_{i'})} \cdot \frac{-\sigma_j}{\prod_{j' \neq j} (\tau_j - \tau_{j'})} \quad \text{and} \quad v_j \leftarrow \tau_j \tag{2.19}$$

(2.18) becomes

$$\left(\sum_{j=1}^{l} \frac{\sigma_j}{\left(\prod_{j' \neq j} \tau_j - \tau_{j'} \right)} \frac{1}{a_i - \tau_j} \ : \ i \in [n] \right) = \left(\frac{\sum_{j=1}^{l} \sigma_j \prod_{j' \neq j} \frac{a_i - \tau_{j'}}{\tau_j - \tau_{j'}}}{\prod_{j=1}^{l} a_i - \tau_j} \ : \ i \in [n] \right) \in \widehat{\mathbb{F}}'^n. \tag{2.20}$$

Since the change of variables is invertible, $p \circ A$ vanishes at (2.18) if and only if it vanishes at (2.20).

Now, viewing $\sigma_1, \ldots, \sigma_l, \tau_1, \ldots, \tau_l$ as seed variables, observe that the right-hand side of (2.20) is $\mathrm{RFE}_l^{l-1}(g/h)$ where g is parametrized by evaluations ($g(\tau_j) = \sigma_j$) and h is parametrized by roots $(\tau_1, \ldots, \tau_l)$. It follows that $p \circ A$ vanishes at (2.20) if and only if $(p \circ A)(\mathrm{RFE}_l^{l-1}) = 0$. The lemma follows. ∎

2.3 Generating Set

In this section we establish Theorem 2.2, the characterization of the vanishing ideal of RFE in terms of an explicit generating set. For every $k, l \in \mathbb{N}$, we develop a template,

EVC_l^k, for constructing polynomials that belong to the vanishing ideal of RFE_l^k such that all instantiations collectively generate the vanishing ideal.

We start by deriving the template. The seeds f of RFE_l^k are of the form $f = g/h$, where $g, h \in \mathbb{F}[\alpha]$ with $\deg(g) \le k$, $\deg(h) \le l$, and $h(a_i) \neq 0$ for each $i \in [n]$. By definition, $\mathrm{RFE}_l^k(f)$ substitutes each variable x_i by $f(a_i) = g(a_i)/h(a_i)$. In particular, the equation $x_i = g(a_i)/h(a_i)$ becomes satisfied for each $i \in [n]$, or, equivalently, $h(a_i)x_i - g(a_i) = 0$. Organizing the coefficients of the monomial expansions of $h(\alpha) = \sum_{d=0}^{l} h_d \alpha^d$ and $g(\alpha) = \sum_{d=0}^{k} g_d \alpha^d$ into column vectors $\vec{h} \doteq \begin{bmatrix} h_l & h_{l-1} & \dots & h_1 & h_0 \end{bmatrix}^\top$ and $\vec{g} \doteq \begin{bmatrix} g_k & g_{k-1} & \dots & g_1 & g_0 \end{bmatrix}^\top$, we can rewrite these equations as the following system of linear equations in the $k + l + 2$ coefficients of g and h combined:

$$\begin{bmatrix} a_i^l x_i & a_i^{l-1} x_i & \dots & x_i & a_i^k & a_i^{k-1} & \dots & 1 \end{bmatrix}_{i \in [n]} \cdot \begin{bmatrix} \vec{h} \\ -\vec{g} \end{bmatrix} = 0. \tag{2.21}$$

Note that the system's coefficient matrix has no dependence on the seed f. Consider any square subsystem of (2.21), formed by choosing $k + l + 2$ indices $i_1, i_2, \dots, i_{k+l+2} \in [n]$ and looking at the corresponding rows. After substitution by $\mathrm{RFE}(f)$ for any fixed seed f, the subsystem has a nonzero solution (namely the vector in (2.21)) and therefore the determinant of its coefficient matrix vanishes.

Before the substitution by $\mathrm{RFE}(f)$, the determinant of the subsystem's coefficient matrix is a polynomial in $x_{i_1}, x_{i_2}, \dots, x_{i_{k+l+2}}$, independent of the seed f:

$$p = \det \begin{bmatrix} a_{i_j}^l x_{i_j} & a_{i_j}^{l-1} x_{i_j} & \dots & x_{i_j} & a_{i_j}^k & a_{i_j}^{k-1} & \dots & 1 \end{bmatrix}_{j=1}^{k+l+2}. \tag{2.22}$$

As p vanishes after substitution of the variables by $\mathrm{RFE}_l^k(f)$ for every seed f, by definition p belongs to the vanishing ideal of RFE_l^k. Recalling that p is identically $\mathrm{EVC}_l^k[i_1, i_2, \dots, i_{k+l+2}]$, we have established:

Claim 2.9. *For every $k, l \in \mathbb{N}$ and $i_1, i_2, \dots, i_{k+l+2} \in [n]$, $\mathrm{EVC}_l^k[i_1, \dots, i_{k+l+2}] \in \mathrm{Van}[\mathrm{RFE}_l^k]$.*

Before moving on, we point out the following properties.

Proposition 2.10. *If any of $i_1, \ldots, i_{k+l+2}$ coincide,* $\mathrm{EVC}_l^k[i_1, \ldots, i_{k+l+2}]$ *is zero. Otherwise, it is nonzero, multi-linear, and homogeneous of total degree $l+1$, and every multi-linear monomial of degree $l+1$ in $x_{i_1}, \ldots, x_{i_{k+l+2}}$ appears with a nonzero coefficient.* EVC_l^k *is skew-symmetric in that, for any permutation π of $i_1, \ldots, i_{k+l+2}$,*

$$\mathrm{EVC}_l^k[i_1, \ldots, i_{k+l+2}] = (-1)^{\mathrm{sign}(\pi)} \cdot \mathrm{EVC}_l^k[\pi(i_1), \ldots, \pi(i_{k+l+2})]. \tag{2.23}$$

The coefficient of $x_{i_1} \cdot \cdots \cdot x_{i_{l+1}}$ is the product of Vandermonde determinants

$$\begin{vmatrix} a_{i_1}^l & \cdots & 1 \\ \vdots & \ddots & \vdots \\ a_{i_{l+1}}^l & \cdots & 1 \end{vmatrix} \begin{vmatrix} a_{i_{l+2}}^k & \cdots & 1 \\ \vdots & \ddots & \vdots \\ a_{i_{l+k+2}}^k & \cdots & 1 \end{vmatrix}. \tag{2.24}$$

Proof. All the assertions to be proved follow from elementary properties of determinants, that Vandermonde determinants are nonzero unless they have duplicate rows, and the following computation: After plugging in 1 for $x_{i_1}, \ldots, x_{i_{l+1}}$, and 0 for $x_{i_{l+2}}, \ldots, x_{i_{l+k+2}}$, the determinant has the form

$$\begin{vmatrix} a_{i_1}^l & \cdots & 1 & * & \cdots & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{i_{l+1}}^l & \cdots & 1 & * & \cdots & * \\ & & & & & \\ 0 & \cdots & 0 & a_{i_{l+2}}^k & \cdots & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{i_{l+k+2}}^k & \cdots & 1 \end{vmatrix}, \tag{2.25}$$

which equals the product of Vandermonde matrices in the statement. ∎

Claim 2.9 shows that the polynomials $\mathrm{EVC}_l^k[i_1, \ldots, i_{k+l+2}]$ belong to the vanishing ideal of RFE_l^k. To prove that they collectively generate the vanishing ideal, we use a two-phase approach:

1. First, we show that every polynomial is equal, modulo the ideal $\langle \mathrm{EVC}_l^k \rangle$ generated by the instantiations of EVC_l^k, to a polynomial with a particular combinatorial structure (Lemma 2.12).

2. Then we show that every nonzero polynomial with that structure is hit by RFE_l^k (Lemma 2.14).

Together, these show that every polynomial in the vanishing ideal of RFE_l^k is equal, modulo $\langle \mathrm{EVC}_l^k \rangle$, to the zero polynomial. It follows that the vanishing ideal is generated by instantiations of EVC_l^k.

The combinatorial structure is that of a core, which is the set C in the following definition.

Definition 2.11 (Cored polynomial). For $c, t \in \mathbb{N}$, a polynomial p is said to be (c,t)-*cored* if there exists a set of at most c variables such that every monomial of p depends on at most t variables outside that set. ◂

Lemma 2.12. *For every $k, l \in \mathbb{N}$, and any $(k+1)$-subset $C \subseteq [n]$, every polynomial is equal to a $(k+1, l)$-cored polynomial with core $\{x_i : i \in C\}$ modulo the ideal generated by the polynomials $\mathrm{EVC}_l^k[S]$ where S ranges over all sets of size $k+l+2$ satisfying $C \subseteq S \subseteq [n]$.*

Proof of Lemma 2.12. Fix k, l, and C as in the statement, and let I be the ideal in the lemma statement. Every monomial m can be uniquely factored as $m_0 m_1$, where m_0 is supported on variables indexed by C and m_1 involves no variable indexed by C. Call m_1 the *non-core* of m. We show the following:

Claim 2.13. *Every monomial with more than l variables in its non-core is equivalent, modulo I, to a linear combination of monomials that all have non-cores of lower degree.*

This lets us prove Lemma 2.12 as follows. Claim 2.13 implies that, for any polynomial p, we may, without changing p mod I, eliminate any monomial in p that violates the $(k+1, l)$-cored condition, while possibly introducing monomials with lower non-core degree. Thus we can systematically eliminate all monomials that violate the cored condition by eliminating them in order of decreasing non-core degree. After that, p is $(k+1, l)$-cored with core $\{x_i : i \in C\}$, and the lemma follows.

It remains to show Claim 2.13. Let m be a monomial with more than l variables in its non-core. Let $L \subseteq [n]$ index a set of $l+1$ of the variables in the non-core, let m' be their product, and let m'' satisfy $m = m'm''$. Combined, L and C have size exactly $k+l+2$. Consider $q \doteq \mathrm{EVC}_l^k[L \cup C]$, where the variables in $L \cup C$ are ordered arbitrarily. By Proposition 2.10, m' appears in q, and every other monomial in q has lower non-core degree than m'. It follows that every monomial in $m'' \cdot q$ either is m, or else has lower non-core degree than m. By the definitions of I and q, $m'' \cdot q$ is in I, so rearranging the equation $m'' \cdot q \equiv 0 \pmod{I}$ to isolate m gives the desired equivalence. ∎

The following lemma completes the proof of the main part of Theorem 2.2, that the polynomials EVC_l^k generate the vanishing ideal of RFE_l^k.

Lemma 2.14. *Suppose p is nonzero and $(k+1,l)$-cored. Then RFE_l^k hits p.*

Before proving Lemma 2.14, let us see how the "moreover" part of Theorem 2.2 also follows. The combination of Claim 2.9, Lemma 2.12, and Lemma 2.14 shows that, for every core $C \subseteq \{x_1, \ldots, x_n\}$ of $k+1$ variables, each instance p of EVC_l^k that does not use all variables in C lies in the ideal generated by those instances that do use all of C. Since all polynomials of the form EVC_l^k have the same degree, this implies that p is actually linearly dependent on the latter instances. Meanwhile, instances of EVC_l^k with distinct variable sets that use all of C are linearly independent, because they each have a distinct monomial. This shows that the instances in the "moreover" part of Theorem 2.2 form a linearly independent set that generate all the instances of EVC_l^k. This completes the proof of Theorem 2.2 modulo the proof of Lemma 2.14.

The proof of Lemma 2.14 involves a key technical analysis that we will repeatedly invoke throughout the chapter. It is abstracted into Lemma 2.17 below. In order to state the lemma, we first define the following notions.

Definition 2.15 (Projection of a polynomial). Let $X \subseteq [n]$, let $p \in \mathbb{F}[x_1, \ldots, x_n]$, and consider the expansion of p as a sum of monomials in $\{x_i : i \in X\}$ with coefficients in

$\mathbb{F}[x_i : i \notin X]$. For any monomial m supported in $\{x_i : i \in X\}$, the *X-projection of p onto m*, denoted $\langle m \,|\, p \rangle_X$, is the coefficient of m in the aforementioned expansion of p. ◂

Note that m need not use every variable indexed by X, and in any case $\langle m \,|\, p \rangle_X$ depends on no variables indexed by X. The notation in Definition 2.15 is inspired by the bra-ket notation from physics.

Definition 2.16 (Minorization). Let $K, L \subseteq [n]$, $p \in \mathbb{F}[x_1, \ldots, x_n]$, and m^* a monomial supported in $\{x_i : i \in K \cup L\}$. We say that m^* is *(K, L)-unminored in p* if, for every monomial m in p, at least one of the following holds:

- $\deg_{x_i}(m) = \deg_{x_i}(m^*)$ for every $i \in K \cup L$,
- $\deg_{x_i}(m) > \deg_{x_i}(m^*)$ for some $i \in K$, or
- $\deg_{x_i}(m) < \deg_{x_i}(m^*)$ for some $i \in L$.

◂

Equivalently, m^* is (K, L)-unminored in p if $\langle m \,|\, p \rangle_{K \cup L} = 0$ for every monomial $m \neq m^*$ supported in $\{x_i : i \in K \cup L\}$ with $\deg_{x_i}(m) \leq \deg_{x_i}(m^*)$ for all $i \in K$, and $\deg_{x_i}(m) \geq \deg_{x_i}(m^*)$ for all $i \in L$.

The above notation enables a succinct statement of this chapter's key technical lemma. We refer to it as the Zoom Lemma because it lets us zoom in on particular monomial parts of the polynomial p, namely the $(K \cup L)$-projection onto a (K, L)-unminored monomial m^*.

Lemma 2.17 (Zoom Lemma). *Let $K, L \subseteq [n]$ be sets of variables. Let $p \in \mathbb{F}[x_1, \ldots, x_n]$, and m^* a monomial supported in $\{x_i : i \in K \cup L\}$ that is (K, L)-unminored in p. If $\langle m^* \,|\, p \rangle_{K \cup L}$ is nonzero at the point*

$$x_i \leftarrow z \cdot \frac{\prod_{i' \in K \setminus L} (a_i - a_{i'})}{\prod_{i' \in L \setminus K} (a_i - a_{i'})} \qquad \forall i \in [n] \setminus (K \cup L), \tag{2.26}$$

for some $z \in \mathbb{F}$ then RFE_l^k hits p with $k = |K|$ and $l = |L|$.

Note that since the $(K \cup L)$-projection of p depends on no variable indexed by $K \cup L$, the result of the substitution (2.26) is simply a scalar in $\mathbb{F}$.

Most of our uses of the Zoom Lemma will moreover have K and L be disjoint, but disjointness is not necessary for the lemma to hold.

Let us first see how the Zoom Lemma allows us complete the proof of Theorem 2.2.

Proof of Lemma 2.14 from the Zoom Lemma. Let $C \subseteq [n]$ denote a core for p. Without loss of generality, C is nonempty. Let M denote the set of monomials with nonzero coefficients in p. Let m_1 be a monomial supported in $\{x_i : i \notin C\}$ that is of maximum degree subject to dividing some monomial of p. Let $L \subseteq [n]$ be the indexes of the variables appearing in m_1; since p is cored, L has size at most l. Fix $i^* \in C$ arbitrarily, and let $K = C \setminus \{i^*\}$; K has size at most k and is disjoint from L. Finally, among the choices for a monomial m_0 supported on K such that $\langle m_0 m_1 \,|\, p \rangle_{K \cup L} \neq 0$, choose one of minimum degree.

The choice of m_1 ensures that every $m \in M$ either has $\deg_{x_i}(m) < \deg_{x_i}(m_1)$ for some $i \in L$, or else $\deg_{x_i}(m) = \deg_{x_i}(m_1)$ for all $i \in L$. In turn, the choice of m_0 ensures that every m in the latter case has $\deg_{x_i}(m) > \deg_{x_i}(m_0)$ for some $i \in K$, or else $\deg_{x_i}(m) = \deg_{x_i}(m_0)$ for all $i \in K$. In the latter of those cases, m must be $m_0 m_1 \cdot x_{i^*}^d$ for some d. In other words, if we set $m^* = m_0 m_1$, then every $m \in M$ satisfies at least one of the following:

- $m = m^* \cdot x_{i^*}^d$ for some d,
- $\deg_{x_i}(m) > \deg_{x_i}(m^*)$ for some $i \in K$, or
- $\deg_{x_i}(m) < \deg_{x_i}(m^*)$ for some $i \in L$.

In particular, m^* is (K, L)-unminored in p, and $\langle m^* \,|\, p \rangle_{K \cup L}$ is a nonzero univariate polynomial in x_{i^*}. It follows that for all but finitely many $z \in \mathbb{F}$, substituting (2.26) into $\langle m^* \,|\, p \rangle_{K \cup L}$ has a nonzero result. By the Zoom Lemma, RFE_l^k hits p. ∎

Finally, we establish the Zoom Lemma. Below we provide a proof from first principles. However, we take intuition from thinking in terms of Laurent expansions, which are

like power series, except that the exponents may go negative. We describe the underlying intuition for readers familiar with the notion.

Consider RFE_l^k in the roots parametrization, where the seed $f = g/h$ is specified by the k roots of the numerator g, the l roots of the denominator h, and an additional scaling parameter ζ (cf. Section 2.2, or (2.27) below). We match each of the roots of g to a unique index in K, and each of the roots of h to a unique index in L. For each $i \in [n]$, $f(a_i)$ is a rational function in the root parameters and ζ, and moreover is a product of univariate rational functions. For each i and root parameter σ with matching index i, we can expand the univariate rational function in σ to its Laurent series about $\sigma = a_i$. Then we carry these expansions into $p(\mathrm{RFE}_l^k(f))$ and expand fully, collecting terms according to the powers of the various $\sigma - a_i$. The result is a multivariate Laurent expansion of $p(\mathrm{RFE}_l^k(f))$ with respect to the root parameters around their matching abscissas, with coefficients that are polynomials in ζ. According to our matching between the root parameters and the variables in K and L, we can index the coefficients in the Laurent expansion of $p(\mathrm{RFE}_l^k)$ by the monomials supported in $\{x_i : i \in K \cup L\}$. The point is that, since m^* is (K, L)-unminored in p, the only contribution to the coefficient indexed by m^* comes from the constant term in the corresponding Laurent expansion of $q(\mathrm{RFE}_l^k)$, where q is the $(K \cup L)$-projection of p onto m^*. Since q does not depend on any variable indexed by $K \cup L$, this Laurent expansion of $q(\mathrm{RFE}_l^k)$ has no terms of negative exponent, so the constant term may be computed by substituting $\sigma \leftarrow a_i$ into $q(\mathrm{RFE}_l^k)$ for each root parameter σ with matching index i. After moreover substituting $\zeta \leftarrow z$, this equals the substitution of (2.26) into q. Since the substitution of (2.26) into q is nonzero, we conclude that $p(\mathrm{RFE}_l^k)$ is a nonzero function of the parameters, and so RFE_l^k hits p.

Proof of the Zoom Lemma. Let ζ, σ_i for each $i \in K$, and τ_i for each $i \in L$ be fresh, distinct indeterminates. Let $\widehat{\mathbb{F}}$ be the field of rational functions in those indeterminates with coefficients in $\mathbb{F}$, and let

$$\hat{f}(\alpha) \doteq \zeta \cdot \frac{\prod_{i \in K}(\alpha - \sigma_i)}{\prod_{i \in L}(\alpha - \tau_i)} \in \widehat{\mathbb{F}}(\alpha). \tag{2.27}$$

Let $\mathrm{RFE}_l^k(\hat{f}) \in \widehat{\mathbb{F}}^n$ be the point $(\hat{f}(a_i) : i \in [n])$, and consider $p(\mathrm{RFE}_l^k(\hat{f})) \in \widehat{\mathbb{F}}$. Any substitution of ζ, σ_i, and τ_i by scalars in $\mathbb{F}$ such that each $\tau_i \notin \{a_1, \ldots, a_n\}$ sends $\hat{f}$ to some f in the domain of RFE_l^k, and $p(\mathrm{RFE}_l^k(\hat{f}))$ to $p(\mathrm{RFE}_l^k(f))$. If $p(\mathrm{RFE}_l^k(\hat{f}))$ is nonzero, then a random such substitution will have nonzero outcome. So if we can show that $p(\mathrm{RFE}_l^k(\hat{f}))$ is nonzero, then p is hit by RFE_l^k.

Recall m^* and z from the lemma statement, and consider the following process Φ. Given an element of $\widehat{\mathbb{F}}$, first multiply it by

$$\frac{\prod_{i \in L}(a_i - \tau_i)^{\deg_{x_i}(m^*)}}{\prod_{i \in K}(a_i - \sigma_i)^{\deg_{x_i}(m^*)}}, \tag{2.28}$$

then cancel any common factors in the numerator and denominator, and then substitute

$$\begin{aligned} \sigma_i &\leftarrow a_i \quad \text{for } i \in K \\ \tau_i &\leftarrow a_i \quad \text{for } i \in L \quad . \\ \zeta &\leftarrow z \end{aligned} \tag{2.29}$$

For any monomial m such that both

$$\begin{aligned} \deg_{x_i}(m) &\geq \deg_{x_i}(m^*) \qquad \text{for all } i \in K, \text{ and} \\ \deg_{x_i}(m) &\leq \deg_{x_i}(m^*) \qquad \text{for all } i \in L, \end{aligned} \tag{2.30}$$

applying Φ to $m(\mathrm{RFE}_l^k(\hat{f}))$ yields a defined result, which is just a scalar in $\mathbb{F}$. (For other monomials the result has a division by zero, but this will not matter.) If any of the inequalities in (2.30) is strict, the result is zero. Otherwise, $m = m^* \cdot m'$ for some monomial m' supported on $\{x_i : i \notin K \cup L\}$, and the result is

$$\left[\prod_{i \in K \cup L} \left(\zeta \cdot \frac{\prod_{i' \in K \setminus \{i\}} a_i - a_{i'}}{\prod_{i' \in L \setminus \{i\}} a_i - a_{i'}} \right)^{\deg_{x_i}(m^*)} \right] \cdot m'(x_i \leftarrow f^*(a_i)) \tag{2.31}$$

where $f^*(\alpha)$ is $\hat{f}(\alpha)$ with each occurrence of σ_i and τ_i substituted by the corresponding a_i, and ζ replaced by z. Note that m' is supported on $\{x_i : i \in [n] \setminus (K \cup L)\}$, and $f^*(a_i)$ is well-defined for all $i \in [n] \setminus (K \cup L)$. Note also that the first factor in (2.31) is nonzero and independent of m'.

Our hypothesis that m^* is (K, L)-unminored in p implies that p is a linear combination of monomials that satisfy (2.30). Thus applying Φ to $p(\mathrm{RFE}_l^k(\hat{f}))$ has a defined output. This output is precisely the first factor in (2.31) times the evaluation of $\langle m^* \,|\, p\rangle_{K\cup L}$ at $(f^*(a_i) : i \in [n] \setminus (K \cup L))$. Meanwhile, $(f^*(a_i) : i \in [n] \setminus (K \cup L))$ is identically (2.26), and we have hypothesized that $\langle m^* \,|\, p\rangle_{K\cup L}$ does not vanish there. It follows that applying Φ to $p(\mathrm{RFE}_l^k(\hat{f}))$ has a nonzero outcome. On the other hand, if $p(\mathrm{RFE}_l^k(\hat{f}))$ were zero, applying Φ would result in zero. It follows that $p(\mathrm{RFE}_l^k(\hat{f})) \neq 0$, proving the lemma. ∎

2.4 Membership Test

In this section we develop the structured membership test for the vanishing ideal $\mathrm{Van}[\mathrm{RFE}_l^k]$ given in Theorem 2.4. We start by observing that it suffices to establish the following simpler version of Theorem 2.4 for the case where p is homogeneous.

Lemma 2.18. *A nonzero homogeneous multi-linear polynomial p in the variables $x_1, \ldots, x_n$ belongs to* $\mathrm{Van}[\mathrm{RFE}_l^k]$ *if and only if both of the following conditions hold:*

1. *The degree of p satisfies $l < \deg(p) < n - k$.*
2. *For all disjoint subsets $K, L \subseteq [n]$ with $|K| = k$ and $|L| = l$, $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ evaluates to zero upon substituting for each $i \in [n] \setminus (K \cup L)$*

$$x_i \leftarrow \frac{\prod_{i' \in K}(a_i - a_{i'})}{\prod_{i' \in L}(a_i - a_{i'})}. \tag{2.32}$$

To see why the general case reduces to the homogeneous case, we make use of the following property, well-known in the context of SV. We include a proof for completeness.

Proposition 2.19. *For any polynomial p, p vanishes at* RFE *if and only if every homogeneous part of p vanishes at* RFE*.*

Proof of Proposition 2.19. For any seed f for RFE and $\zeta \in \mathbb{F}$, $\zeta \cdot f$ is another seed for RFE over the extended field $\mathbb{F}(\zeta)$ of rational functions in ζ. Write $p = \sum_d p_d$ as a sum of

homogeneous polynomials, where p_d has degree d. Since p_d is homogeneous, $p_d(\mathrm{RFE}(\zeta \cdot f)) = \zeta^d \cdot p_d(\mathrm{RFE}(f))$. Thus for every f,

$$p(\mathrm{RFE}(\zeta \cdot f)) = \sum_d p_d(\mathrm{RFE}(\zeta \cdot f)) = \sum_d \zeta^d \cdot p_d(\mathrm{RFE}(f)) \tag{2.33}$$

is a polynomial $q_f(\zeta) \in \mathbb{F}[\zeta]$. If, for all f and d, $p_d(\mathrm{RFE}(f)) = 0$, then q_f is the zero polynomial for all f, so $p(\mathrm{RFE}(f)) = q_f(1) = 0$ for all f. Conversely, if $p(\mathrm{RFE}(z \cdot f)) = q_f(z) = 0$ for all f and $z \in \mathbb{F}$, then q_f is the zero polynomial for all f, so $p_d(\mathrm{RFE}(f)) = 0$ for all f and d. ∎

Here is how Theorem 2.4 follows from Lemma 2.18.

Proof of Theorem 2.4 from Lemma 2.18. Write $p = \sum_d p_d$ as a sum of homogeneous parts. By Proposition 2.19, $p \in \mathrm{Van}[\mathrm{RFE}_l^k]$ if and only if every $p_d \in \mathrm{Van}[\mathrm{RFE}_l^k]$. The degree constraints in Lemma 2.18 show that condition 1 in Theorem 2.4 is necessary. Thus, in order to establish Theorem 2.4, we only need to consider polynomials p for which $p_d = 0$ for $d \le l$ and $d \ge n-k$, and show that for any such p, all the evaluations (2.7) of p are zero if and only if for all d, all the evaluations (2.32) of p_d are zero.

Fix K, L, Z as in the statements of Theorem 2.4 and Lemma 2.18, and let $Y = [n] \setminus (K \cup L)$. Let $\lambda \in \mathbb{F}^Y$ be the point (2.32), and for $z \in Z$, let $z \cdot \lambda$ denote the point (2.7). We claim $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ vanishes at $z\lambda$ for all $z \in Z$ if and only if $\left(\frac{\partial p_d}{\partial L}\right)\Big|_{K \leftarrow 0}$ vanishes at λ for all d. Let ζ be an indeterminate. We have

$$\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}(\zeta\lambda) = \sum_{l < d < n-k} \left(\frac{\partial p_d}{\partial L}\right)\Big|_{K \leftarrow 0}(\zeta\lambda) = \sum_{l < d < n-k} \zeta^{d-l} \left(\frac{\partial p_d}{\partial L}\right)\Big|_{K \leftarrow 0}(\lambda). \tag{2.34}$$

This is a polynomial in ζ, say $q(\zeta)$. Evaluating q at $\zeta \leftarrow z$ coincides with evaluating $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ at $z\lambda$, while the coefficient of ζ^{d-l} coincides with evaluating $\left(\frac{\partial p_d}{\partial L}\right)\Big|_{K \leftarrow 0}$ at λ. q factors as $\zeta \cdot q'$ where q' has degree at most $n-k-l-2$. Therefore q vanishes on any fixed set of at least $n-k-l-1$ nonzero field elements—in particular Z—if and only if it is the zero polynomial. Theorem 2.4 follows. ∎

It remains to prove Lemma 2.18. For this we once again make use of the Zoom Lemma (Lemma 2.17). Note that for multi-linear polynomials and disjoint K and L, $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ coincides with the projection $\langle m^* \,|\, p \rangle_{K \cup L}$ where $m^* = \prod_{i \in L} x_i$. Moreover, since p is multi-linear, the condition that m^* be (K, L)-unminored in p is automatically satisfied: the only multi-linear monomial m supported in $K \cup L$ with $\deg_{x_i}(m) \leq \deg_{x_i}(m^*)$ for all $i \in K$ and $\deg_{x_i}(m) \geq \deg_{x_i}(m^*)$ for all $i \in L$ is $m = m^*$. This leads to the following specialization of the Zoom Lemma for multi-linear polynomials with disjoint K and L:

Lemma 2.20. *Let $K, L \subseteq [n]$ be disjoint, and let $p \in \mathbb{F}[x_1, \ldots, x_n]$ be a multi-linear polynomial. If $\langle \prod_{i \in L} x_i \,|\, p \rangle_{K \cup L}$ is nonzero at the point*

$$x_i \;\leftarrow\; z \cdot \frac{\prod\limits_{i' \in K} (a_i - a_{i'})}{\prod\limits_{i' \in L} (a_i - a_{i'})} \qquad \forall i \in [n] \setminus (K \cup L), \tag{2.35}$$

for some $z \in \mathbb{F}$ then RFE_l^k hits p with $k = |K|$ and $l = |L|$.

In proving Lemma 2.18, we will apply Lemma 2.20 only to homogeneous polynomials, in which case we can take $z = 1$ without loss of generality. With that in mind, observe that (2.32) in Lemma 2.18 coincides with the substitution (2.35) from Lemma 2.20. So Lemma 2.18 amounts to saying that a homogeneous multi-linear polynomial p is hit by RFE_l^k if and only if its degree is too low, its degree is too high, or else there is a way to apply Lemma 2.20 to prove that p is hit by RFE_l^k.

Proof of Lemma 2.18. Suppose that $\deg(p) \leq l$. Set L to be the indices of the variables appearing in some monomial with nonzero coefficient in p, and set $K \leftarrow \varnothing$. $\langle \prod_{i \in L} x_i \,|\, p \rangle_{K \cup L}$ is a nonzero constant. Lemma 2.20 applies, concluding that RFE_l^0, and hence RFE_l^k, hits p.

Suppose now that $\deg(p) \geq n - k$. Set K to be the indices of the variables *not* appearing in some monomial with nonzero coefficient in p, and set $L \leftarrow \varnothing$. $\langle 1 \,|\, p \rangle_{K \cup L}$ is a single monomial, namely the product of the variables indexed by $[n] \setminus (K \cup L)$. Lemma 2.20 applies. Since none of the substitutions in (2.35) is zero, we conclude that RFE_0^k, and hence RFE_l^k, hits p.

Now consider the case $l < \deg(p) < n-k$. We start by writing p as a multi-linear element of $\mathrm{Van}[\mathrm{RFE}_l^k]$ plus a structured remainder term. It can be shown similarly to Lemma 2.12; we include a proof below.

Claim 2.21. *Let $l < d < n-k$. Every homogeneous degree-d multi-linear polynomial can be written as $p_0 + r$ where p_0 and r are degree-d homogeneous multi-linear polynomials, $p_0 \in \mathrm{Van}[\mathrm{RFE}_l^k]$ and r is $(d+k-l, l)$-cored.*

Let p_0, r be the result of applying the claim to p. By the contrapositive of Lemma 2.20, it holds that for every pair of disjoint subsets $K, L \subseteq [n]$ of sizes k and l respectively, the projection $\langle \prod_{i\in L} x_i \,|\, p_0 \rangle_{K\cup L}$ evaluates to zero at (2.32). Since $\langle \prod_{i\in L} x_i \,|\, p \rangle_{K\cup L} = \langle \prod_{i\in L} x_i \,|\, p_0 \rangle_{K\cup L} + \langle \prod_{i\in L} x_i \,|\, r \rangle_{K\cup L}$, it follows that evaluating $\langle \prod_{i\in L} x_i \,|\, p \rangle_{K\cup L}$ at (2.32) has the same result as evaluating $\langle \prod_{i\in L} x_i \,|\, r \rangle_{K\cup L}$. In light of this, Lemma 2.18 follows from the following claim, proven below:

Claim 2.22. *Let $l < d < n-k$. Let r be a nonzero degree-d homogeneous multi-linear polynomial that is $(d+k-l, l)$-cored. There are disjoint sets $K, L \subseteq [n]$ with $|K| = k$ and $|L| = l$ so that $\langle \prod_{i\in L} x_i \,|\, r \rangle_{K\cup L}$ is a single monomial.*

Substituting (2.32) into a single monomial yields a nonzero value. ∎

We complete the argument by proving Claims 2.21 and 2.22. Claim 2.21 is similar to Lemma 2.12, and is obtained using a variant of polynomial division suited to multi-linear polynomials:

Proof of Claim 2.21. Let $C \subseteq [n]$ have size $d+k-l$. Every multi-linear monomial m can be uniquely factored as $m_0 m_1$, where m_0 and m_1 are multi-linear monomials supported in $\{x_i : i \in C\}$ and $\{x_i : i \notin C\}$ respectively. Call m_1 the *non-core* of m. We show the following:

Claim 2.23. *Every multi-linear monomial with more than l variables in its non-core is equivalent, modulo a multi-linear element of $\mathrm{Van}[\mathrm{RFE}_l^k]$, to a linear combination of multi-linear monomials that all have non-cores of lower degree.*

This lets us prove Claim 2.21 as follows. Claim 2.23 implies that, for any multi-linear polynomial p, we may, without changing p modulo multi-linear elements of $\mathrm{Van}[\mathrm{RFE}_l^k]$, eliminate any monomial in p that violates the $(d+k-l,l)$-cored condition, while possibly introducing multi-linear monomials with lower non-core degree. Thus we can systematically eliminate all monomials that violate the cored condition by eliminating them in order of decreasing non-core degree. After that, p is $(d+k-l,l)$-cored (with core $\{x_i : i \in C\}$), and Claim 2.21 follows.

We now show Claim 2.23. Factor $m = m_0 m_1$ as above, and suppose there are more than l variables in m_1. Let L index some $l+1$ of the variables in m_1, let m' be their product, and let m'' satisfy $m = m'm''$. There are at most $d-l-1$ variables in m_0; let K be any $k+1$ elements of C that index variables not in m_0. Combined, L and K have size exactly $k+l+2$. Consider $q = \mathrm{EVC}_l^k[L \cup K]$, where the variables in $L \cup K$ are ordered arbitrarily. By Proposition 2.10, m' appears as a monomial in q; moreover, every other monomial in q has lower non-core degree. It follows that every monomial in $m'' \cdot q$ either is m, or else has lower non-core degree. Moreover, every such monomial is multi-linear and is supported in $\{x_i : i \in K \cup L\}$, which is disjoint from the support of m''. As q is in $\mathrm{Van}[\mathrm{RFE}_l^k]$, rearranging the equation $m'' \cdot q \equiv 0 \pmod{\mathrm{Van}[\mathrm{RFE}_l^k]}$ to isolate m gives the desired equivalence. ∎

Claim 2.22 is similar to the proof of Lemma 2.14:

Proof of Claim 2.22. Let $C \subseteq [n]$ be the indices of variables that form a core for r. Recall that $l < d < n-k$. By shrinking C if need be, we can assume there is a multi-linear monomial m with nonzero coefficient in r that involves exactly l variables not indexed by C. Let L be the variables appearing in m that are not indexed by C. Now extend C to have size $d+k-l$ while remaining disjoint from L. There are precisely k variables indexed by C that do not appear in m; let K be this set. Since r is multi-linear, homogeneous of degree d, and $(d+k-l,l)$-cored with core C, there is exactly one monomial with nonzero coefficient in r that is divisible by $\prod_{i \in L} x_i$ and by no variable in K: it is precisely m. It follows that the

projection $\langle \prod_{i \in L} x_i \,|\, r \rangle_{K \cup L}$ is a single monomial. ■

We conclude this section by detailing the connection between Theorem 2.4 and some prior applications of the SV-generator.

Application to Read-Once Formulas We start with the theorem that SV^1 hits read-once formulas. The original proof in [MV18] goes by induction on the depth of F, showing that $F(\mathrm{SV}^1)$ is nonconstant whenever F is nonconstant, or, equivalently, that SV^1 hits $F + c$ for every $c \in \mathbb{F}$ whenever F is nonconstant. The inductive step consists of two cases, depending on whether the top gate is a multiplication gate or an addition gate. The case of a multiplication gate follows from the general property that the product of a nonconstant polynomial with any nonzero polynomial is nonconstant. The case of an addition gate, say $F = F_1 + F_2$, involves a clever analysis that uses the variable-disjointness of F_1 and F_2 to show that $F_1(\mathrm{SV}^1)$ and $F_2(\mathrm{SV}^1)$ cannot cancel each other out.

The case of an addition gate $F = F_1 + F_2$ alternately follows from Theorem 2.4 with $k = 0$ and $l = 1$ and the following two observations, each corresponding to one of the conditions in Theorem 2.4. Both observations are immediate because of the variable-disjointness of F_1 and F_2:

1. If at least one of F_1 of F_2 has a homogeneous component of degree 1 or at least n, then so does F.

2. If for $L = \{i\} \subseteq [n]$ at least one of the derivatives $\frac{\partial F_1}{\partial x_i}$ or $\frac{\partial F_2}{\partial x_i}$ is nonzero at some point (2.7), then the same goes for $\frac{\partial F}{\partial x_i}$.

In particular, under the hypothesis that $F_1 + c$ is hit by RFE_1^0 for all $c \in \mathbb{F}$, F_1 must violate one of the conditions of Theorem 2.4 besides the one that requires F_1 have no constant term. Similarly for F_2. By the above observations, any such a violation is inherited by F, and the inductive step follows.

As mentioned in the overview, Theorem 2.4 was originally proved from a perspective that carries a geometric interpretation. The case of an addition gate in the above proof takes a particularly clean form in that perspective, which we shall see now.

Recall from the overview that we can think of the variables as vertices, and multi-linear monomials simplices made from those vertices. A multi-linear polynomial is a weighted collection of such simplices with weights from $\mathbb{F}$. In this view, Theorem 2.4 translates to the following characterization: a weighted collection of simplices corresponds to a polynomial in the vanishing ideal of RFE_1^0 if and only if there are no simplices of zero, one, or all vertices, and the remaining weights satisfy a certain system of linear equations. Crucially, for each equation in the system, there is a vertex such that the equation reads only weights of the simplices *that contain that vertex*. Meanwhile, the sum of two variable-disjoint polynomials corresponds to taking the vertex-disjoint union of two weighted collections of simplices. It follows directly that if either term in the sum violates a requirement besides the "no simplex of zero vertices" requirement, then the sum violates the same requirement.

Zero-Substitutions and Partial Derivatives As mentioned in the overview, several prior papers demonstrated the utility of partial derivatives and zero substitutions in the context of derandomizing PIT using the SV-generator, especially for syntactically multi-linear models. By judiciously choosing variables for those operations, these papers managed to simplify p and reduce PIT for p to PIT for simpler instances, resulting in an efficient recursive algorithm. Such recursive arguments can be naturally reformulated to use Theorem 2.4, according to the following prototype.

Let $\mathcal{C}$ be a family of multi-linear polynomials, such as those computable with some bounded complexity in some syntactic model. For the argument, we break up $\mathcal{C} = \bigcup_{k,l} \mathcal{C}_{k,l}$ such that for every k, l and $p \in \mathcal{C}_{k,l}$, at least one of the following holds:

- $k = l = 0$ and p is either zero or hit by RFE_0^0.
- $k > 0$ and there is a zero substitution such that the result is in $\mathcal{C}_{k-1,l}$.

- $l > 0$ and there is a derivative such that the result is in $\mathcal{C}_{k,l-1}$.

We also make the mild assumption that each $\mathcal{C}_{k,l}$ is closed under rescaling variables. With these hypotheses in place, we establish the following claim through direct applications of Theorem 2.4:

Claim 2.24. *Under the above hypotheses,* RFE_l^k *hits* $\mathcal{C}_{k,l}$ *for every* k, l.

Proof. The proof is by induction on k and l. The base case is $k = l = 0$, where the claim is immediate. When $k > 0$ or $l > 0$, our hypotheses are such that p either simplifies under a zero substitution $x_{i^*} \leftarrow 0$ or a derivative $\frac{\partial}{\partial x_{i^*}}$. We analyze each case separately. By condition 1 of Theorem 2.4, we may assume that p only has homogeneous parts with degrees in the range $l+1, \dots, n-k-1$.

- If p simplifies under a zero substitution $x_{i^*} \leftarrow 0$, then let $p' \in \mathcal{C}_{k-1,l}$ be the simplified polynomial where moreover the remaining variables have been rescaled according to $x_i \leftarrow x_i \cdot (a_{i^*} - a_i)$. That is, write p as $p = qx_{i^*} + r$ where q and r are polynomials that do not depend on x_{i^*}, and set $p'(\dots, x_i, \dots) \doteq r(\dots, x_i \cdot (a_{i^*} - a_i), \dots)$. By induction, p' is hit by RFE_l^{k-1}. We apply Theorem 2.4 to p' with respect to the set of variables $\{x_1, \dots, x_{i^*-1}, x_{i^*+1}, \dots, x_n\}$ and k replaced by $k-1$. As p only has homogeneous parts with degrees in the range $l+1, \dots, n-k-1$, so does p', and condition 1 of Theorem 2.4 fails. By condition 2, there must be $z \in Z$ and disjoint $K, L \subseteq [n] \setminus \{i\}$ with $|K| = k-1$ and $|L| = l$ so that substituting (2.7) yields a nonzero value. It follows directly that, with respect to the same z, $K' = K \cup \{i\}$, and the same L, the substitution (2.7) yields a nonzero value when applied to p.

- If p simplifies under a partial derivative $\frac{\partial}{\partial x_{i^*}}$, then a similar analysis works. Set $p' \in \mathcal{C}_{k,l-1}$ to be the simplification with variables rescaled according to $x_i \leftarrow x_i/(a_{i^*} - a_i)$. That is, write p as $p = qx_{i^*} + r$ where q and r are polynomials that do not depend on x_{i^*}, and set $p'(\dots, x_i, \dots) \doteq q(\dots, x_i/(a_{i^*} - a_i), \dots)$. By induction, p' is hit by RFE_{l-1}^k. We

apply Theorem 2.4 to p' with respect to the set of variables $\{x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_n\}$ and l replaced by $l-1$. As p' has homogeneous parts of degrees one less than p does, condition 1 of Theorem 2.4 fails. By condition 2, there is $z \in Z$ and disjoint $K, L \subseteq [n] \setminus \{i\}$ with $|K| = k$ and $|L| = l-1$ so that substituting (2.7) yields a nonzero value. It follows directly that, with respect to the same z, the same K, and $L' = L \cup \{i^*\}$, the substitution (2.7) yields a nonzero value when applied to p. ∎

2.5 Sparseness

By Proposition 2.10, the generators EVC_l^k contain exactly $\binom{k+l+2}{l+1}$ monomials. The following result shows that no nonzero polynomial in the vanishing ideal of RFE_l^k has fewer monomials.

Lemma 2.25. *Suppose $p \in \mathbb{F}[x_1, \ldots, x_n]$ is nonzero and has only s monomials with nonzero coefficients. Then, for any k, l such that $\binom{k+l+2}{l+1} > s$, RFE_l^k hits p.*

The tactic here is to show that if p has too few monomials appearing in it, then there is necessarily a way to instantiate the Zoom Lemma where $\langle m^* \,|\, p \rangle_X$ is just a single monomial, and therefore nonzero at one of the requisite points in the Zoom Lemma.

Proof. Let M be a set of monomials. For $i \in [n]$, let $\downarrow_i$ be the operation mapping M to its subset consisting of those monomials in which x_i appears at its lowest degree. Let $\uparrow_i$ be similar, except we select the monomials in which x_i appears at its highest degree. We make the following claim:

Claim 2.26. *For any nonempty set of monomials with fewer than $\binom{k+l+2}{l+1}$ monomials, there is a sequence of $\downarrow$ and $\uparrow$ operations, with at most k $\downarrow$ operations and at most l $\uparrow$ operations, such that the resulting set of monomials has exactly one element.*

The claim implies the lemma as follows. Let M be the set of monomials with nonzero coefficient in p. Apply the claim to M to get a sequence of $\downarrow$ and $\uparrow$ operations resulting in a single monomial m_0. Let K denote the set of variables used for the $\downarrow$ operations, L the set of

variables used for the $\uparrow$ operations, and $X = K \cup L$. Let m^* be the monomial supported on $\{x_i : i \in X\}$ where each variable x_i appears with same degree as in m_0. By how the operators are defined, every monomial m in M satisfies either

- $\deg_{x_i}(m) > \deg_{x_i}(m^*)$ for some $i \in K$ (m was removed by $\downarrow_{x_i}$),
- $\deg_{x_i}(m) < \deg_{x_i}(m^*)$ for some $i \in L$ (m was removed by $\uparrow_{x_i}$), or
- $\deg_{x_i}(m) = \deg_{x_i}(m^*)$ for every $i \in K \cup L$, in which case $m = m_0$.

Therefore, m^* is (K, L)-unminored in p and the Zoom Lemma applies. $\langle m^* \,|\, p \rangle_X$ is a single monomial, so it does not vanish at the required point. We conclude that p is hit by $\mathrm{RFE}_{|L|}^{|K|}$, and therefore by RFE_l^k.

It remains to prove Claim 2.26. We do this by induction on $|M|$. In the base case, $|M| = 1$, in which case the empty sequence suffices. Otherwise, $|M| > 1$, in which case there is a variable x_i that appears with at least two different degrees in monomials in M. The sets $\downarrow_i (M)$ and $\uparrow_i (M)$ are nonempty and disjoint. Since M has size less than $\binom{k+l+2}{l+1} = \binom{k+l+1}{l+1} + \binom{k+l+1}{l}$, either $\downarrow_i (M)$ has size less than $\binom{k+l+1}{l+1}$, or $\uparrow_i (M)$ has size less than $\binom{k+l+1}{l}$. Whichever is the case, the claim follows by applying the inductive hypothesis to it. ∎

2.6 Set-Multi-Linearity

Although the generators EVC_l^k provided by Theorem 2.2 are not set-multi-linear, the vanishing ideal of RFE_l^k does contain set-multi-linear polynomials. In this section, we construct some of degree $l + 1$ with partition classes of size $k + 2$. In fact, we argue that all set-multi-linear polynomials in $\mathrm{Van}[\mathrm{RFE}_l^k]$ of degree $l+1$ are in the linear span of the ones we construct, and conclude that no polynomial of degree $l + 1$ in $\mathrm{Van}[\mathrm{RFE}_l^k]$ can be set-multi-linear with partitions of size less than $k + 2$.

The construction is a modification of the one for EVC_l^k.

Definition 2.27. Let $k, l \in \mathbb{N}$ be parameters, and let $X_1, \ldots, X_{l+1} \subseteq [(l+1)(k+2)]$ be $l+1$ disjoint subsets of $k+2$ variables each. The polynomial ESMVC_l^k is an $(l+1) \times (l+1)$ determinant where each entry is itself a $(k+2) \times (k+2)$ determinant. We index the rows in the outer determinant by $i = 1, \ldots, l+1$, and the columns by $d = l, \ldots, 0$. In each (i,d)-th inner matrix, there is one row per $j \in X_i$; it is

$$\begin{bmatrix} a_j^d x_j & a_j^k & a_j^{k-1} & \cdots & a_j^1 & a_j^0 \end{bmatrix}. \tag{2.36}$$

◂

The name "ESMVC" is a shorthand for "Elementary Set-Multi-linear Vandermonde Circulation".

Similar to EVC, the precise instantiation of ESMVC requires one to pick an order for the sets $X_1, \ldots, X_{l+1}$ (up to even permutations) and an order within each set (again up to even permutations). Changing any of those orders by an odd permutation causes the sign of ESMVC to flip, but otherwise it is unchanged.

Example 2.28. When $k = 1$ and $l = 2$, ESMVC uses three sets of three variables each. To help convey the structure of the determinant, we name the variable-sets $\{x_1, x_2, x_3\}$, $\{y_1, y_2, y_3\}$, and $\{z_1, z_2, z_3\}$, and denote the abscissa of x_i by a_i, the abscissa of y_i by b_i, and

the abscissa of z_i by c_i. With this notation, ESMVC is the following:

$$\left\|\begin{vmatrix} a_1^2x_1 & a_1^1 & a_1^0 \\ a_2^2x_2 & a_2^1 & a_2^0 \\ a_3^2x_3 & a_3^1 & a_3^0 \end{vmatrix} \begin{vmatrix} a_1^1x_1 & a_1^1 & a_1^0 \\ a_2^1x_2 & a_2^1 & a_2^0 \\ a_3^1x_3 & a_3^1 & a_3^0 \end{vmatrix} \begin{vmatrix} a_1^0x_1 & a_1^1 & a_1^0 \\ a_2^0x_2 & a_2^1 & a_2^0 \\ a_3^0x_3 & a_3^1 & a_3^0 \end{vmatrix}\right\| \cdot \left\|\begin{vmatrix} b_1^2y_1 & b_1^1 & b_1^0 \\ b_2^2y_2 & b_2^1 & b_2^0 \\ b_3^2y_3 & b_3^1 & b_3^0 \end{vmatrix} \begin{vmatrix} b_1^1y_1 & b_1^1 & b_1^0 \\ b_2^1y_2 & b_2^1 & b_2^0 \\ b_3^1y_3 & b_3^1 & b_3^0 \end{vmatrix} \begin{vmatrix} b_1^0y_1 & b_1^1 & b_1^0 \\ b_2^0y_2 & b_2^1 & b_2^0 \\ b_3^0y_3 & b_3^1 & b_3^0 \end{vmatrix}\right\| \cdot \left\|\begin{vmatrix} c_1^2z_1 & c_1^1 & c_1^0 \\ c_2^2z_2 & c_2^1 & c_2^0 \\ c_3^2z_3 & c_3^1 & c_3^0 \end{vmatrix} \begin{vmatrix} c_1^1z_1 & c_1^1 & c_1^0 \\ c_2^1z_2 & c_2^1 & c_2^0 \\ c_3^1z_3 & c_3^1 & c_3^0 \end{vmatrix} \begin{vmatrix} c_1^0z_1 & c_1^1 & c_1^0 \\ c_2^0z_2 & c_2^1 & c_2^0 \\ c_3^0z_3 & c_3^1 & c_3^0 \end{vmatrix}\right\|. \tag{2.37}$$

◂

Proposition 2.29. *For any $k, l \geq 0$ and variable-index sets $X_1, \ldots, X_{l+1}$ as in Definition 2.27,* ESMVC *is nonzero, homogeneous of degree $l+1$, and set-multi-linear with respect to the partition $X_1 \sqcup \cdots \sqcup X_{l+1}$. Moreover, every monomial consistent with that appears with a nonzero coefficient.* ESMVC *is skew-symmetric with respect to the order of the sets $X_1, \ldots, X_{l+1}$, and the choice of order within each set, in that any permutation thereof changes the construction by merely multiplying by the sign of the permutation. When the sets are ordered as $X_1, \ldots, X_{l+1}$ and their members are ordered as $X_i = \{x_{i,1}, \ldots, x_{i,k+2}\}$ for $i = 1, \ldots, l+1$, the coefficient of $x_{1,1} \cdot \cdots \cdot x_{l+1,1}$ is the product of Vandermonde determinants*

$$\begin{vmatrix} a_{1,1}^l & \cdots & a_{1,1}^0 \\ \vdots & \ddots & \vdots \\ a_{l+1,1}^l & \cdots & a_{l+1,1}^0 \end{vmatrix} \cdot \prod_{i=1}^{l+1} \begin{vmatrix} a_{i,2}^k & \cdots & a_{i,2}^0 \\ \vdots & \ddots & \vdots \\ a_{i,k+2}^k & \cdots & a_{i,k+2}^0 \end{vmatrix}. \tag{2.38}$$

Proof. All assertions to be proved follow from elementary properties of determinants, that Vandermonde determinants are nonzero unless they have duplicate rows, and the following computation: the result of plugging 1 into $x_{i,1}$ for $i = 1, \ldots, l+1$ and 0 into the remaining variables is the product of Vandermonde matrices in the statement. ∎

The following theorem formalizes the role ESMVC plays among the degree-$(l+1)$ elements of $\mathrm{Van}[\mathrm{RFE}_l^k]$.

Theorem 2.30. *Let $k, l \in \mathbb{N}$ be parameters, and let $X_1, \ldots, X_{l+1}$ be $l+1$ disjoint sets of variable-indices (of any size). Let $\mathrm{ESMVC}_l^k(X_1, \ldots, X_{l+1})$ be the collection of polynomials formed by picking a $(k+2)$-subset of each of $X_1, \ldots, X_{l+1}$ and instantiating ESMVC_l^k with respect to those sets. The linear span of $\mathrm{ESMVC}_l^k(X_1, \ldots, X_{l+1})$ equals the set-multi-linear polynomials in $\mathrm{Van}[\mathrm{RFE}_l^k]$ with variable partition $X_1 \sqcup \cdots \sqcup X_{l+1}$.*

Theorem 2.30 immediately implies that there are no set-multi-linear polynomials of degree $l+1$ in $\mathrm{Van}[\mathrm{RFE}_l^k]$ that have at least one partition X_i of size less than $k+2$.

Proving Theorem 2.30 involves two steps, similar to Theorem 2.2:

1. Show that any instantiation of ESMVC_l^k is in $\mathrm{Van}[\mathrm{RFE}_l^k]$.
2. Show that, modulo instantiations of ESMVC_l^k, every set-multi-linear polynomial with variable partition $X_1, \ldots, X_{l+1}$ takes a particular form such that RFE_l^k hits every nonzero polynomial of that form.

Step 1 is the following claim:

Claim 2.31. *For every $k, l \in \mathbb{N}$, and every choice of $l+1$ disjoint sets $X_1, \ldots, X_{l+1}$ of $k+2$ variable-indices each, ESMVC_l^k vanishes at RFE_l^k.*

Proof. Let g/h be a seed for RFE_l^k. Let A be the $(l+1) \times (l+1)$ outer matrix defining ESMVC, so that $\mathrm{ESMVC} \doteq \det(A)$. Recall that the columns of A are indexed by $d = l, \ldots, 0$. Let $h \in \mathbb{F}^{l+1}$ be the column vector where the row indexed by d is the coefficient of α^d in

$h(\alpha)$. We show that, after substituting $\mathrm{RFE}_l^k(g/h)$, the matrix-vector product $Ah \in \mathbb{F}^{l+1}$ yields the zero vector. It follows that evaluating ESMVC at $\mathrm{RFE}_l^k(g/h)$ vanishes, as it is the determinant of a singular matrix.

Fix $i \in \{1, \ldots, l+1\}$, and focus on the i-th coordinate of Ah. The (i,d) entry of A is a determinant; let $B_{i,d}$ be the inner matrix as in Definition 2.27. As d varies, only the first column of $B_{i,d}$ changes. Thus, by multi-linearity of the determinant, the i-th entry of Ah is itself a determinant. Recalling that the rows of $B_{i,l}, \ldots, B_{i,0}$ are indexed by $j \in X_i$, the j-th row of this determinant is

$$\begin{bmatrix} h(a_j)x_j & a_j^k & \cdots & a_j^0 \end{bmatrix}. \tag{2.39}$$

After substituting $\mathrm{RFE}_l^k(g/h)$, it becomes

$$\begin{bmatrix} g(a_j) & a_j^k & \cdots & a_j^0 \end{bmatrix}. \tag{2.40}$$

Since g is a degree-k polynomial, the columns are linearly dependent, so the determinant is zero. ■

For step 2, we need a suitable replacement for being (c,t)-cored. The following adaptation of that to the set-multi-linear setting suffices.

Definition 2.32. Let $X_1 \sqcup \cdots \sqcup X_d \subseteq [n]$ be disjoint sets of variable-indices. For parameters $k, l \geq 0$, a polynomial p that is set-multi-linear with respect to the sets $X_1, \ldots, X_d$ is (c,t)-*multi-cored* if there are subsets $C_i \subseteq X_i$ for $i = 1, \ldots, d$, each of size at most c, such that every monomial of p involves at most l variables not indexed by $C_1 \cup \cdots \cup C_d$. ◂

Claim 2.33. *Let $k, l \geq 0$ be parameters, and let $X_1 \sqcup \cdots \sqcup X_{l+1} \subseteq [n]$ be disjoint sets of variable-indices. Let $\mathrm{ESMVC}_l^k(X_1, \ldots, X_{l+1})$ be the collection of polynomials formed by picking a $(k+2)$-subset of each of $X_1, \ldots, X_{l+1}$ and instantiating ESMVC_l^k with respect to those sets. Every set-multi-linear polynomial with variable partition $X_1, \ldots, X_{l+1}$ equals a $(k+1, l)$-multi-cored polynomial modulo the linear span of $\mathrm{ESMVC}_l^k(X_1, \ldots, X_{l+1})$.*

Claim 2.33 follows from a monomial elimination argument as in Lemma 2.12. A formal proof is omitted. From there, Theorem 2.30 follows from the following claim:

Claim 2.34. *Let $k, l \geq 0$ be parameters, and let $X_1 \sqcup \cdots \sqcup X_{l+1} \subseteq [n]$ be disjoint sets of variable-indices. Every degree-$(l+1)$ polynomial that is set-multi-linear with respect to the partition $X_1, \ldots, X_{l+1}$ and that is $(k+1, l)$-multi-cored is hit by* RFE_l^k.

Proof. Let p satisfy the hypotheses of the claim, and let $C_1, \ldots, C_{l+1}$ be the sets witnessing the $(k+1, l)$-multi-core structure on p. Every monomial m factors as $m_0 m_1$ with m_0 supported on $\{x_i : i \in C_1 \cup \cdots \cup C_{l+1}\}$ and m_1 supported on $\{x_i : i \notin C_1 \cup \cdots \cup C_{l+1}\}$. Call m_1 the *non-core* of m. Fix $m = m_0 m_1$ to be a monomial in p so that the non-core is maximal under divisibility. The multi-core structure on p implies the non-core has at most $l < l+1$ variables, so m_0 contains at least one variable, x_j. Let i such that $x_j \in X_i$. Refactor $m = x_j \cdot m^*$. Let L index the variables in m^* and set $K = C_i \setminus \{j\}$. The set-multi-linear structure implies that $\langle m^* \,|\, p \rangle_{K \cup L}$ uses only variables indexed by X_i. By how we chose m and the multi-core structure, it must use variables indexed by C_i. By how we chose K, it must use only x_j. It follows that $\langle m^* \,|\, p \rangle_{K \cup L}$ is just a nonzero scalar times x_j. The Zoom Lemma applies, and we conclude that $p \notin \mathrm{Van}[\mathrm{RFE}_l^k]$. ∎

2.7 Read-Once Oblivious Arithmetic Branching Programs

In this section we provide some background on ROABPs and establish Theorem 2.5.

2.7.1 Background

Algebraic branching programs are a syntactic model for algebraic computation. One forms a directed graph with a designated source and sink. Each edge is labeled by a polynomial that depends on at most one variable among $x_1, \ldots, x_n$. The branching program computes

a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ by summing, over all source-to-sink paths, the product of the labels on the edges of each path.

A special subclass of algebraic branching programs are read-once oblivious algebraic branching programs (ROABPs). In this model, the vertices of the branching program are organized in layers. The layers are totally ordered, and edges exist only from one layer to the next. For each variable, there is at most one consecutive pair of layers between which that variable appears, and for each pair of consecutive layers, there is at most one variable that appears between them. In this way, every source-to-sink path reads each variable at most once (the branching program is *read-once*), and the order in which the variables are read is common to all paths (the branching program is *oblivious*). We can always assume that the number of layers equals one plus the number of variables under consideration.

The number of vertices comprising a layer is called its *width*. The width of an ROABP is the largest width of its layers. The minimum width of an ROABP computing a given polynomial can be characterized in terms of the rank of coefficient matrices constructed as follows.

Definition 2.35. Let $U \sqcup V = [n]$ be a partition of the variable indices, and let M_U and M_V be the sets of monomials that are supported on variables indexed by U and V, respectively. For any polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ define the matrix

$$\mathrm{CMat}_{U,V}(p) \in \mathbb{F}^{M_U \times M_V} \tag{2.41}$$

by setting the (m_U, m_V) entry to equal the coefficient of $m_U m_V$ in p. ◂

$\mathrm{CMat}_{U,V}(p)$ is formally an infinite matrix, but it has only finitely many nonzero entries. When p has degree at most d, one can just as well truncate $\mathrm{CMat}_{U,V}(p)$ to include only rows and columns indexed by monomials of degree at most d.

Lemma 2.36 ([Nis91]). *Let $p \in \mathbb{F}[x_1, \ldots, x_n]$ be any polynomial. There is an ROABP of width w computing p in the variable order $x_1, \ldots, x_n$ if and only if, for every $s \in \{0, \ldots, n\}$,*

with respect to the partition $U = \{1, \ldots, s\}$ *and* $V = \{s+1, \ldots, n\}$*, we have*

$$\operatorname{rank}(\mathrm{CMat}_{U,V}(p)) \leq w. \tag{2.42}$$

Lemma 2.36 applies to other variable orders by renaming the variables.

We group the monomials in M_U and M_V by their degrees, and order the groups by increasing degree. This induces a block structure on $\mathrm{CMat}_{U,V}(p)$ with one block for every choice of $r, c \in \mathbb{N}$; the (r, c) block is the submatrix with rows indexed by degree-r monomials in M_U and columns indexed by degree-c monomials in M_V. In the case where p is homogeneous, the only nonzero blocks occur for $r + c$ equal to the degree of p. In this case the rank of $\mathrm{CMat}_{U,V}(p)$ is the sum of the ranks of its blocks.

In general, the rank of $\mathrm{CMat}_{U,V}(p)$ is at least the rank of $\mathrm{CMat}_{U,V}(p_\downarrow)$, where $p_\downarrow$ denotes the homogeneous part of p of the lowest degree, $d_\downarrow$. This follows because the submatrix of $\mathrm{CMat}_{U,V}(p)$ consisting of the rows and columns indexed by monomials of degree at most $d_\downarrow$ has a block structure that is triangular with the blocks of $\mathrm{CMat}_{U,V}(p_\downarrow)$ on the hypotenuse. The observation yields the following folklore consequence of Lemma 2.36.

Proposition 2.37. *Let* $p \in \mathbb{F}[x_1, \ldots, x_n]$ *be any nonzero polynomial, and let* $p_\downarrow$ *be the nonzero homogeneous part of* p *of least degree. If* p *can be computed by an ROABP of width* w*, then so can* $p_\downarrow$*.*

2.7.2 Proof of Theorem 2.5

We now prove that SV^l, or equivalently RFE_l^{l-1}, hits every polynomial computed by an ROABP of width less than $1 + (l/3)$ that contains a monomial of degree at most $l + 1$ (Theorem 2.5). The novelty lies in the special case where p is homogeneous of degree $l + 1$ and multi-linear.

Theorem 2.38. *Let* $l \geq 1$ *a parameter. For any nonzero, multi-linear, homogeneous polynomial* p *of degree* $l + 1$*, if* p *is computable by an ROABP of width less than* $1 + (l/3)$*, then* RFE_l^{l-1} *hits* p*.*

Theorem 2.5 follows from Theorem 2.38 in a standard way. We provide a proof for completeness.

Proof of Theorem 2.5 from Theorem 2.38. Fix p satisfying the hypotheses of Theorem 2.5. We show that RFE_l^{l-1} hits p; this implies SV^l hits p because RFE_l^{l-1} and SV^l are equivalent up to variable rescaling, and rescaling variables does not affect ROABP width. Let $p_\downarrow$ be the nonzero homogeneous part of p of least degree. We show that RFE_l^{l-1} hits $p_\downarrow$; this implies RFE_l^{l-1} hits p by Proposition 2.19.

Suppose first that $p_\downarrow$ contains a monomial m^* depending on at most l variables. It is well-known that SV^l hits any such polynomial. Here is an argument based on the Zoom Lemma. Set $K = \varnothing$ and L to be the set of variables appearing in m^* The homogeneity of $p_\downarrow$ ensures that m^* is (K, L)-unminored. Meanwhile, the $(K \cup L)$-projection of p onto m^* is a nonzero constant. By the Zoom Lemma, RFE_l^{l-1} hits $p_\downarrow$.

Since $\deg(p_\downarrow) \le \deg(p) \le l+1$, the remaining possibility is that $p_\downarrow$ is multi-linear of degree exactly $l+1$. By Proposition 2.37, $p_\downarrow$ is computable by an ROABP of width less than $1+(l/3)$. That RFE_l^{l-1} hits $p_\downarrow$ then follows from Theorem 2.38. ∎

In the remainder of this section we establish Theorem 2.38. Toward this end, fix $l \ge 1$ and fix a variable order. Without loss of generality, the variable order is $x_1, \ldots, x_n$. Our strategy is to show that, for every p that is nonzero, multi-linear, and homogeneous of degree $l+1$, and that moreover belongs to the vanishing ideal of RFE_l^{l-1}, there is some $s \in \{0, \ldots, n\}$ so that, with respect to the partition $U = \{1, \ldots, s\}$, $V = \{s+1, \ldots, n\}$, it holds that $\mathrm{rank}(\mathrm{CMat}_{U,V}(p)) \ge 1 + (l/3)$. This suffices to prove Theorem 2.38 by Lemma 2.36.

Let $C \doteq \mathrm{CMat}_{U,V}(p)$. As p is multi-linear, we only need to consider monomials of the form $\prod_{i \in I} x_i$ for $I \subseteq U$ as row indices, and monomials of the form $\prod_{j \in J} x_j$ for $J \subseteq V$ as column indices. This allows us to index rows by subsets $I \subseteq U$ and columns by subsets $J \subseteq V$.

For $d \in \{0, \ldots, l+1\}$, let C_d denote the $(d, l+1-d)$ block of C, i.e., the submatrix of C corresponding to the rows indexed by subsets $I \subseteq U$ of size $|I| = d$, and the columns indexed

by subsets $J \subseteq V$ of size $|J| = l + 1 - d$. As p is homogeneous of degree $l + 1$, those are the only blocks that can be nonzero, and $\operatorname{rank}(C) = \sum_{d=0}^{l+1} \operatorname{rank}(C_d)$.

If many of the blocks C_d are nonzero, it immediately follows that $\operatorname{rank}(C)$ is high. Thus, we can focus on cases where a lot of the blocks C_d are zero. In that case we'll make use of linear equations given by Lemma 2.18 and, more generally, the Zoom Lemma to argue that, for an appropriate choice of the partition index s, $\operatorname{rank}(C)$ is high.

First consider Lemma 2.18 applied to p. Note that the evaluation of $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ at the points (2.32) in condition 2 can be expressed as a linear function of the coefficients of p. As the coefficients of p are the entries of C, Lemma 2.18 gives us a homogeneous linear equation in the entries of C for each choice of K and L. Note also that, since $|L| = l$ and p is homogeneous of degree $l + 1$, the only monomials m that contribute to $\left(\frac{\partial p}{\partial L}\right)\Big|_{K \leftarrow 0}$ are of the form $m = x_i \cdot \prod_{j \in L} x_j$ for some $i \in \overline{K \cup L}$. It follows that the only entries of C that appear in the corresponding linear equation reside in the two adjacent blocks $C_{|L \cap U|+1}$ (for $i \in U$) and $C_{|L \cap U|}$ (for $i \in V$). As a consequence, we obtain the following:

Proposition 2.39. *Let $p \in \operatorname{Van}[\operatorname{RFE}_l^{l-1}]$ be multi-linear, and homogeneous of degree $l + 1$, let $U \sqcup V$ be a partition of $[n]$, and let $C \doteq \operatorname{CMat}_{U,V}(p)$. Suppose that for some $d_1, d_2 \in \{0, \ldots, l+1\}$ with $d_1 \leq d_2$, $C_{d_1} = 0$ and $C_{d_2} = 0$. Then setting the blocks C_d with $d < d_1$ or $d > d_2$ to zero and keeping all the other blocks the same yields another polynomial $p' \in \operatorname{Van}[\operatorname{RFE}_l^{l-1}]$.*

Proof. We only need to consider the case where p' is nonzero. It suffices to show that whenever p satisfies the two conditions in Lemma 2.18, then so does p'.

Condition 1 holds for p' as it holds for p and p' has the same degree as p.

Consider the equations in condition 2 of Lemma 2.18 for the membership of p' to $\operatorname{Van}[\operatorname{RFE}_l^{l-1}]$. Those that only involve block C_d with $d \leq d_1$ are met as the equations are homogeneous and the blocks involved are all zero. The same holds for the equations that only involve blocks C_d with $d \geq d_2$. The remaining equations only involve blocks C_d with $d \in \{d_1, \ldots, d_2\}$, on which p and p' agree; since those equations hold for p, they also hold for p'. ∎

We will also make use of the following specific linear equations that follow from the Zoom Lemma. For $I \subseteq U$ and $J \subseteq V$ we denote by $C(I, J)$ the entry of C in the row indexed by I (corresponding to the monomial $\prod_{i \in I} x_i$) and the column indexed by J (corresponding to the monomial $\prod_{j \in J} x_j$).

Lemma 2.40. *Let $p \in \mathrm{Van}[\mathrm{RFE}_l^{l-1}]$ be multi-linear, and homogeneous of degree $l+1$, let $U \sqcup V$ be a partition of $[n]$, and let $C \doteq \mathrm{CMat}_{U,V}(p)$. For every $I \subseteq U$ and $J \subseteq V$ with $|I| + |J| = l$, and for every $i^* \in I \cup J$,*

$$\sum_{i \in U \setminus I} \frac{C(\{i\} \cup I, J)}{a_i - a_{i^*}} + \sum_{i \in V \setminus J} \frac{C(I, \{i\} \cup J)}{a_i - a_{i^*}} = 0. \tag{2.43}$$

Proof. Set $L \doteq I \cup J$, $K \doteq L \setminus \{i^*\}$, and $m^* \doteq \prod_{i \in L} x_i$. As p is multi-linear, it follows that m^* is (K, L)-unminored in p. Since $p \in \mathrm{Van}[\mathrm{RFE}_l^{l-1}]$, the contrapositive of the Zoom Lemma tells us that the $(K \cup L)$-projection of p onto m^* vanishes at the point (2.26) for each $z \in \mathbb{F}$. We will use $z = 1$.

The multi-linear monomials m of degree $l+1$ with nonzero $(K \cup L)$-projection onto m^* are of the form $m = m^* \cdot x_i$ for $i \in [n] \setminus (K \cup L)$. Thus, we can write the $(K \cup L)$-projection of p onto m^* as

$$\langle m^* \,|\, p \rangle_{K \cup L} = \sum_{i \in U \setminus I} C(\{i\} \cup I, J) \cdot x_i + \sum_{i \in V \setminus J} C(I, \{i\} \cup J) \cdot x_i. \tag{2.44}$$

Note that the component x_i corresponding to $i \in [n] \setminus (K \cup L)$ in the evaluation point (2.26) for $z = 1$ equals $1/(a_i - a_{i^*})$. Plugging the latter expression into our expression (2.44) for $\langle m^* \,|\, p \rangle_{K \cup L}$, the above application of the Zoom Lemma yields equation (2.43). ∎

Recall that we can focus on the case where a lot of the blocks C_d are zero. The equations given by Lemma 2.40 allow us to derive a good lower bound on $\mathrm{rank}(C)$ in case all the extreme blocks (those with d close to zero or close to $l+1$) all vanish.

Lemma 2.41. *Let $p \in \mathrm{Van}[\mathrm{RFE}_l^{l-1}]$ be nonzero, multi-linear, and homogeneous of degree $l+1$, let $U \sqcup V$ be a partition of $[n]$, and let $C \doteq \mathrm{CMat}_{U,V}(p)$. If every monomial in p*

depends on at least d^* variables indexed by U and at least d^* variables indexed by V, then $\operatorname{rank}(C) \geq d^* + 1$.

Before proving Lemma 2.41, we first show how it allows us to prove Theorem 2.38. Combining Lemma 2.41 with Proposition 2.39 and the simple rank bound based on the number of nonzero blocks yields the following:

Claim 2.42. *Suppose that, for some $d \in [l]$, $C_{d-1} = 0$ and $C_d \neq 0$. Then* $\operatorname{rank}(C) \geq \min(l - 2d + 3, d + 1)$.

Proof. We consider two cases.

Case 1: $C_{d'} \neq 0$ for each $d' \in \{d+1, \ldots, l-d+2\}$.

As $C_d \neq 0$ also holds, C has at least $l - 2d + 3$ nonzero blocks, so $\operatorname{rank}(C) \geq l - 2d + 3$.

Case 2: There exists $d' \in \{d+1, \ldots, l-d+2\}$ such that $C_{d'} = 0$.

Consider the polynomial p' that agrees with p on the blocks $C_{d''}$ with $d'' \in \{d, \ldots, d'-1\}$ and has all its other blocks zero. By Proposition 2.39 with $d_1 = d - 1$ and $d_2 = d'$, $p' \in \operatorname{Van}[\operatorname{RFE}_l^{l-1}]$. The polynomial p' is nonzero since its d-th block is nonzero. It is homogeneous of degree $l + 1$ and multi-linear as all of its monomials also occur in the homogeneous multi-linear polynomial p of degree $l+1$. By construction, every monomial in p' contains at least d variables indexed by U, and at least $l + 1 - (d' - 1) = l - d' + 2 \geq d$ variables indexed by V. Thus, p' satisfies the conditions of Theorem 2.38 with $d^* = d$. It follows that $\operatorname{rank}(C) \geq \operatorname{rank}(\operatorname{CMat}_{U,V}(p')) \geq d + 1$.

As at least one of the two cases applies, we conclude that $\operatorname{rank}(C) \geq \min(l - 2d + 3, d + 1)$. ∎

The best lower bound on $\operatorname{rank}(C)$ that Claim 2.42 can possibly give is $\max_{d \in [l]} \min(l - 2d + 3, d + 1)$. Note that the expression $l - 2d + 3$ is decreasing with d, whereas $d + 1$ is increasing. It follows that the maximum of $\min(l - 2d + 3, d + 1)$ over all real values of d is achieved when $l - 2d + 3 = d + 1$, i.e., for $d = (l + 2)/3$. For $d = \left\lfloor \frac{l+2}{3} \right\rfloor$, the increasing term $d + 1$ is binding, so $\min(l - 2d + 3, d + 1) = d + 1 = 1 + \left\lfloor \frac{l+2}{3} \right\rfloor \geq 1 + (l/3)$. Thus, we obtain the lower bound of

Theorem 2.38 provided we can realize the hypothesis of Claim 2.42 for $d = \left\lfloor \frac{l+2}{3} \right\rfloor$. For this we use our flexibility in the choice of s to select the partition $U = \{1, \ldots, s\}, V = \{s+1, \ldots, n\}$.

Claim 2.43. *For every $d \in [l]$, there is $s \in \{0, \ldots, n\}$ such that, with respect to the partition $U = \{1, \ldots, s\}$, $V = \{s+1, \ldots, n\}$, $C_{d-1} = 0$ and $C_d \neq 0$.*

Proof. When $s = 0$, C_0 contains all entries. As s increases by 1, some entries move from their current block $C_{d'}$ to the next block $C_{d'+1}$. Finally, when $s = n$, C_{l+1} contains all entries. It follows that every nonzero entry moves from C_{d-1} to C_d at some time. If we stop increasing s right after the last nonzero entry of C moves out of C_{d-1}, we have that $C_{d-1} = 0$ and $C_d \neq 0$. ∎

All that remains is to establish Lemma 2.41.

Proof of Lemma 2.41. The proof goes by induction on d^*. The base case is $d^* = 0$, where the lemma holds because the rank of a nonzero matrix is always at least 1. For the inductive step, where $d^* \geq 1$, we make use of the single-variable projections of p. More precisely, for $i^* \in [n]$, let p_{i^*} denote the $\{i^*\}$-projection of p onto x_{i^*}, i.e., the unique polynomial p_{i^*} such that $p = p_{i^*} x_{i^*} + r$ for some polynomial r, where neither p_{i^*} nor r depend on x_{i^*}. Note that for any i^*, p_{i^*} is multi-linear and homogeneous of degree l, and every monomial in p_{i^*} depends on at least $d^* - 1$ variables indexed by U and at least $d^* - 1$ variables indexed by V. In a moment, we argue that $p_{i^*} \in \mathrm{Van}[\mathrm{RFE}_{l-1}^{l-2}]$. Then we will show the following:

Claim 2.44. *There exists $i^* \in [n]$ such that $p_{i^*} \neq 0$ and*

$$\mathrm{rank}(\mathrm{CMat}_{U,V}(p_{i^*})) \leq \mathrm{rank}(\mathrm{CMat}_{U,V}(p)) - 1. \tag{2.45}$$

Given an i^* as in Claim 2.44, we conclude by induction that

$$\mathrm{rank}(\mathrm{CMat}_{U,V}(p)) \geq \mathrm{rank}(\mathrm{CMat}_{U,V}(p_{i^*})) + 1 \geq (d^* - 1) + 1 + 1 = d^* + 1. \tag{2.46}$$

To see that p_{i^*} belongs to the vanishing ideal of RFE_{l-1}^{l-2}, we use Lemma 2.18. Condition 1 of Lemma 2.18 is satisfied by p_{i^*} since it it is satisfied by p, and all of k, l, and the degree

of p_{i^*} are one less. Given K and L as in condition 2 of Lemma 2.18, we have

$$\left(\frac{\partial p'}{\partial L}\right)\Big|_{K\leftarrow 0} = \langle x_{i^*} \cdot \textstyle\prod_{i\in L} x_i \,|\, p\rangle_{K\cup L\cup\{i^*\}}\,. \tag{2.47}$$

Since $p \in \mathrm{Van}[\mathrm{RFE}_l^{l-1}]$, the contrapositive of the Zoom Lemma applied to p with $K' = K \cup \{i^*\}$, $L' = L \cup \{i^*\}$, $m^* = \prod_{i\in L} x_i$, and $z = 1$, says that (2.47) vanishes at (2.32). So $p' \in \mathrm{Van}[\mathrm{RFE}_{l-1}^{l-2}]$ by Lemma 2.18.

It remains to prove Claim 2.44. Let $U' \subseteq U$ be the indices of variables x_i such that p depends on x_i, and similarly define $V' \subseteq V$. Our objective is to find i^* belonging to $U' \cup V'$ and so that (2.45) holds. We first consider the possibility that (2.45) fails for every $i^* \in V'$. We show that this can only happen when $|V'| < |U'|$. A symmetric argument shows that if (2.45) fails for all $i^* \in U'$, then it must be that $|U'| > |V'|$. Both inequalities cannot simultaneously occur; this guarantees the existence of the desired i^*, and the lemma follows as discussed above.

Suppose that (2.45) fails for each $i^* \in V'$. Observe that the column of $\mathrm{CMat}_{U,V}(p_{i^*})$ corresponding to a monomial m equals the column of $\mathrm{CMat}_{U,V}(x_{i^*} \cdot p_{i^*})$ corresponding to the monomial $x_{i^*} \cdot m$; all other columns of $\mathrm{CMat}_{U,V}(x_{i^*} \cdot p_{i^*})$ are zero. The matrix $\mathrm{CMat}_{U,V}(x_{i^*} \cdot p_{i^*})$ can also be formed from $\mathrm{CMat}_{U,V}(p)$ by zeroing out all the columns indexed by subsets that do not contain i^* (corresponding to multi-linear monomials not involving x_{i^*}). The failure of (2.45) implies that $\mathrm{CMat}_{U,V}(p_{i^*})$ has the same rank as $\mathrm{CMat}_{U,V}(p)$, which is to say that the columns of $\mathrm{CMat}_{U,V}(p)$ indexed by subsets that contain i^* span *all* the columns of $\mathrm{CMat}_{U,V}(p)$. Going block by block, this implies that for every block C_d of $C = \mathrm{CMat}_{U,V}(p)$, the columns within C_d that are indexed by subsets containing i^* span all the columns of C_d. This goes for every $i^* \in V'$, since we are supposing that (2.45) fails for every such i^*.

Now let d be minimal such that $C_d \neq 0$, i.e., such that p has a monomial depending on exactly d variables indexed by U. Note that $d \geq d^* \geq 1$, and $C_{d-1} = 0$. The entries of C_d appear in the linear equations (2.43) given in Lemma 2.40, either with entries from C_{d-1} or from C_{d+1}. Since C_{d-1} is zero, some of the equations involving C_{d-1} and C_d simplify to equations on C_d only. Namely, for every $I \subseteq U$ with $|I| = d-1$, every $J \subseteq V$ with $|J| = l-(d-1)$,

and every $i^* \in I \cup J$, equation (2.43) simplifies to

$$\sum_{i \in U \setminus I} \frac{C_d(\{i\} \cup I, J)}{a_i - a_{i^*}} = 0. \tag{2.48}$$

Note that for any fixed $i \in U \setminus U'$, all entries of the form $C_d(\{i\} \cup I, J)$ are zero. Thus, we can restrict the range of i in (2.48) from $U \setminus I$ to $U' \setminus I$:

$$\sum_{i \in U' \setminus I} \frac{C_d(\{i\} \cup I, J)}{a_i - a_{i^*}} = 0. \tag{2.49}$$

Since $C_d \neq 0$, for at least one fixed I not all entries of the form $C_d(\{i\} \cup I, J)$ are zero. Let I^* be such an I, and let C_d^* denote the submatrix of C_d consisting of all entries of the form $C_d(\{i\} \cup I^*, J)$. It follows that for every $J \subseteq V$ with $|J| = l - (d-1)$ and every $i^* \in I^* \cup J$, it holds that

$$\sum_{i \in U' \setminus I^*} \frac{C_d^*(\{i\} \cup I^*, J)}{a_i - a_{i^*}} = 0. \tag{2.50}$$

We make use of the equations (2.50) for $i^* \in V'$ where J ranges over all subsets of V of size $|J| = l - (d-1)$ that contain i^*. Note that, for fixed i^*, the coefficients $\frac{1}{a_i - a_{i^*}}$ in (2.50) are independent of the choice of J. We argued that the columns of C_d indexed by subsets J that contain i^*, span all columns of C_d. The same holds for C_d^* as C_d^* is obtained from C_d by removing rows. It follows that (2.50) holds for *every* subset J of V of size $l - (d-1)$ (not just the ones containing i^*).

In particular, let $J^* \subseteq V$ with $|J^*| = l - (d-1)$ be such that the column $C_d^*(\{i\} \cup I^*, J^*)$ is nonzero (where i ranges over $U' \setminus I^*$). Such J^* exists by the construction of I^*. The column represents a nontrivial solution to the homogeneous system (2.50) of $|V'|$ linear equations (one for each choice of $i^* \in V'$) in $|U' \setminus I^*|$ unknowns (one for each $i \in U' \setminus I^*$). The coefficient matrix $\left[\frac{1}{a_i - a_{i^*}}\right]$ is a Cauchy matrix, which is well-known to have full rank. It follows that the number of equations is strictly less than the number of unknowns, so $|V'| < |U' \setminus I^*| \leq |U'|$. ■

2.8 Alternating Algebra Representation

In this section we present in greater detail the alternating algebra-based representation of polynomials suited to studying the vanishing ideal of RFE.

Per Proposition 2.19, RFE acts separately on the homogeneous parts of any polynomial, so we focus on homogeneous polynomials. We use d as the parameter for degree. Nonzero polynomials with $d \leq l$ are automatically outside the ideal, leaving the case of $d = l + 1$ as the simplest nontrivial case. Subsection 2.8.1 expands the informal discussion in the introduction, describing the representation and characterization for this case where moreover $l = 1$ and $k = 0$. With that in hand, Subsection 2.8.2 provides a brief introduction to alternating algebra suited to our purpose, and then Subsection 2.8.3 formalizes the discussion in Subsection 2.8.1 and extends it to the case of general k and l.

2.8.1 Basic case

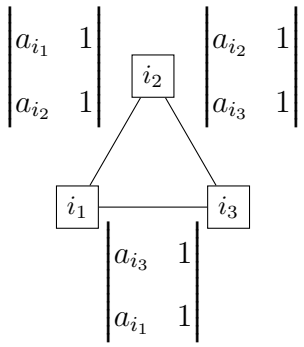
Let $\{x_1, \ldots, x_n\}$ be a set of variables. For the purposes of this subsection, we fix the parameters $k = 0$, $l = 1$, and $d = 2$. That is to say, we are studying which degree-2 polynomials belong to the vanishing ideal for RFE_1^0. When $d = l+1$, polynomials that are not multi-linear are automatically outside the ideal by an application of the Zoom Lemma (cf. the proof of Theorem 2.5 from Theorem 2.38), so we focus on multi-linear polynomials.

In Theorem 2.2, we proved that the polynomials $\mathrm{EVC}_1^0[i_1, i_2, i_3]$ as i_1, i_2, i_3 range over $[n]$ generate $\mathrm{Van}[\mathrm{RFE}_1^0]$. As these generators are all degree-2 polynomials, a degree-2 polynomial is in the ideal if and only if it is a linear combination of instantiations of EVC_1^0. Consider the generator when expanded as a linear combination of monomials:

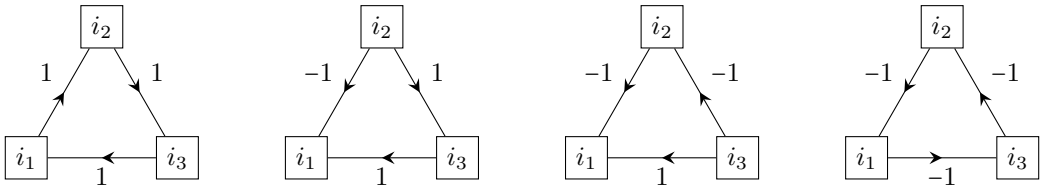
$$\mathrm{EVC}_1^0[i_1, i_2, i_3] = \begin{vmatrix} a_{i_1} & 1 \\ a_{i_2} & 1 \end{vmatrix} x_{i_1} x_{i_2} + \begin{vmatrix} a_{i_3} & 1 \\ a_{i_1} & 1 \end{vmatrix} x_{i_3} x_{i_1} + \begin{vmatrix} a_{i_2} & 1 \\ a_{i_3} & 1 \end{vmatrix} x_{i_2} x_{i_3}. \tag{2.51}$$

We may represent it graphically by making a vertex for each variable, an undirected edge for each monomial, and assigning to each edge a weight equal to the coefficient of that

monomial:



Observe that the coefficient of $x_{i_1}x_{i_2}$ has no dependence on a_{i_3}. In particular, as i_3 varies, the coefficient of $x_{i_1}x_{i_2}$ in $\mathrm{EVC}_1^0[i_1, i_2, i_3]$ does not change. In any other instantiation of EVC_1^0 involving both i_1 and i_2, the coefficient is either the same, or else differs by a sign. Similar holds with respect to all other monomials. This suggests to modify the graphical representation by rescaling the weights on edges. To account for the signs, we orient the edges. More precisely, for each edge $\{i_1, i_2\}$, we select an arbitrary orientation for it, say $i_1 \to i_2$, and then divide its coefficient by $\begin{vmatrix} a_{i_1} & 1 \\ a_{i_2} & 1 \end{vmatrix}$. With these changes, $\mathrm{EVC}_1^0[i_1, i_2, i_3]$ may be drawn in any of the following ways (among others):

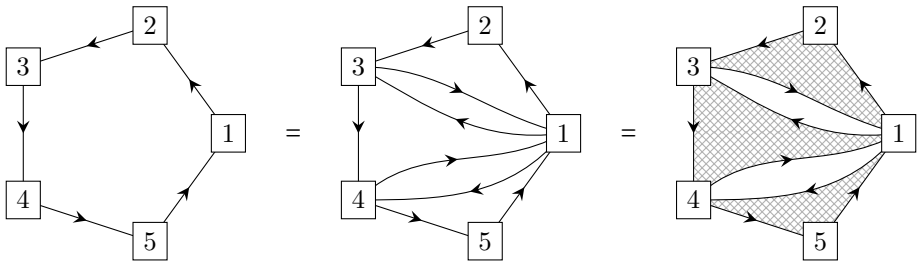


In general, for any degree-2 homogeneous multi-linear polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$, we can represent it in a similar way: create a vertex i for each variable x_i, and create and arbitrarily orient an edge for each monomial. For each edge $i_1 \to i_2$, set its coefficient to be the coefficient of $x_{i_1}x_{i_2}$ in p divided by $\begin{vmatrix} a_{i_1} & 1 \\ a_{i_2} & 1 \end{vmatrix}$. Note this graphical representation is linear in the polynomial: adding or rescaling polynomials coincides with adding or rescaling coefficients on the edges. Moreover, the representation determines the polynomial: simply undo the scaling on each edge, and read off a linear combination of monomials.

Observe that in the graphical representation of $\mathrm{EVC}_1^0[i_1, i_2, i_3]$, at every vertex, the sum of the coefficients on edges oriented out of that vertex equals the sum of the coefficients on edges oriented in to that vertex. This is true no matter how we orient the edges. Indeed, we can interpret $\mathrm{EVC}_1^0[i_1, i_2, i_3]$ as a *circulation* in which one unit of flow travels around a simple 3-cycle $i_1 \to i_2 \to i_3 \to i_1$. The coefficient on an oriented edge $i_1 \to i_2$ measures how much flow is traveling in the direction $i_1 \to i_2$, with negatives representing flow in the opposite direction. That the sum of coefficients on outgoing edges equals the sum of coefficients on incoming edges means that this circulation satisfies a *conservation law* at every vertex: the total flow in equals the total flow out.

Since every degree-2 polynomial in $\mathrm{Van}[\mathrm{RFE}_1^0]$ is a linear combinations of instantiations of EVC_1^0, each is represented by a circulation *that also satisfies the conservation law*. Thus conservation is a necessary condition for membership in $\mathrm{Van}[\mathrm{RFE}_1^0]$. Not every polynomial satisfies this condition: consider, for example, any lone monomial, or a sum of variable-disjoint monomials.

Indeed, conservation is *sufficient* for ideal membership as well. It is folklore that every circulation that satisfies conservation can be decomposed as a linear combination of unit circulations around simple cycles. Unit circulations around simple cycles can be decomposed as a sum of unit circulations on 3-cycles; this is depicted for a 5-cycle below, where each edge indicates unit flow:



(2.52)

The basis of the first equality above is that a unit flow $i_1 \to i_2$ cancels with a unit flow $i_2 \to i_1$.

In summary, a homogeneous degree-2 multi-linear polynomial is in $\mathrm{Van}[\mathrm{RFE}_1^0]$ *if and only if*, when represented as a circulation, it satisfies the conservation law. This is the representation and ideal membership characterization for such polynomials in the special

case $k = 0$, $l = 1$, and $d = 2$.

2.8.2 Alternating algebra

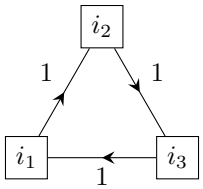
In order to generalize Subsection 2.8.1, we need to be able to discuss higher-dimensional analogues of "flow" and "circulation", as well as appropriately-generalized notions of "conservation". Suited to this purpose is the language of *alternating algebra*. Alternating algebra was introduced in the 1800s by Hermann Grassmann [Gra44; GK00], and is the formalism underlying differential geometry and its applications to physics. We give only a brief introduction to alternating algebra here, tailored heavily toward our purposes.

For each variable x_i, we create for it with a fresh vertex, labeled i. The alternating algebra provides a multiplication, denoted $\wedge$, that can be thought of as a constructor to make *oriented simplices* out of these vertices. For example, the $\wedge$-product of i_1 with i_2, written $i_1 \wedge i_2$, encodes the simplex with vertices i_1 and i_2. When $i_1 = i_2$, $i_1 \wedge i_2$ is defined to be zero. $\wedge$-multiplication is associative. Rather than being commutative, the $\wedge$-product is *anticommutative* in the sense that $i_1 \wedge i_2 = -i_2 \wedge i_1$. In this way the order of the vertices in the product encodes an orientation. There are only ever two orientations: in a larger product such as $i_1 \wedge i_2 \wedge i_3$, we have

$$\begin{aligned} i_1 \wedge i_2 \wedge i_3 &= -i_1 \wedge i_3 \wedge i_2 \\ = \; i_3 \wedge i_1 \wedge i_2 &= -i_3 \wedge i_2 \wedge i_1 \\ = \; i_2 \wedge i_3 \wedge i_1 &= -i_2 \wedge i_1 \wedge i_3. \end{aligned} \tag{2.53}$$

In general, permuting the vertices in a $\wedge$-product by an even permutation has no effect, while permuting by an odd permutation flips the sign. Any $\wedge$-product that uses the same vertex more than once is zero. The alternating algebra consists of all formal linear combinations of $\wedge$-products of vertices formed in the preceding way. This includes as a distinct simplex the empty product, denoted 1, which is an identity for $\wedge$. The $\wedge$-product distributes over addition.

To connect this with Subsection 2.8.1, recall the graphical depiction of $\mathrm{EVC}_1^0[i_1, i_2, i_3]$:



Adopting the convention that an arrow $i_1 \to i_2$ is $i_1 \wedge i_2$ (and so an arrow $i_2 \to i_1$ is $i_2 \wedge i_1 = -i_1 \wedge i_2$), we can alternatively express the above as

$$i_1 \wedge i_2 \; + \; i_2 \wedge i_3 \; + \; i_3 \wedge i_1. \tag{2.54}$$

In general, the graphical representation of a degree-2 multi-linear polynomial is some linear combination of such 2-vertex simplices. When we go to higher-degree polynomials, we will use simplices with more vertices.

To express conservation, we use *boundary maps*. Denoted by ∂_w where $w : [n] \to \mathbb{F}$ is any function, they are linear maps that send each simplex to a linear combination of its boundary faces (and the empty simplex to zero) according to the following formula:

$$\partial_w(i_1 \wedge \cdots \wedge i_r) = \sum_{j=1}^{r} (-1)^{1+j} w(i_j)\; i_1 \wedge \cdots \wedge i_{j-1} \wedge i_{j+1} \wedge \cdots \wedge i_r. \tag{2.55}$$

The map ∂_w extends linearly to all elements of the alternating algebra. In the simplest case, w is the constant-1 function. In this case, the boundary of some 2-vertex simplex is given by

$$\partial_w(i_1 \wedge i_2) = i_1 - i_2. \tag{2.56}$$

In particular, $i_1 \wedge i_2$ contributes $+1$ toward i_1 and -1 toward i_2. This coincides with the contribution of the edge $i_1 \to i_2$ toward the net flow out of the vertices i_1 and i_2. In exactly this way, conservation is identified with having a *vanishing boundary*. Different choices of w give rise to different boundary operators.

As we generalize parameters, it will become important to iterate boundary operators. The first fact here is that taking the same boundary multiple times always vanishes. That is, for any w, $\partial_w \circ \partial_w = 0$. The second is that for any w, w' and $\beta, \beta' \in \mathbb{F}$, $\partial_{\beta w + \beta' w'} = \beta \partial_w + \beta' \partial_{w'}$,

which is to say that the boundary operators themselves are linear in w. It follows from these that, for any w, w', $\partial_w \circ \partial_{w'} = -\partial_{w'} \circ \partial_w$. This means that the boundary operators themselves behave like an alternating algebra, with $\circ$ as the multiplication rather than $\wedge$. For any $w_1, \ldots, w_k$, write $\omega = w_1 \wedge \cdots \wedge w_k$, and define $\partial_\omega = \partial_{w_k} \circ \cdots \circ \partial_{w_1}$. That is, $w_1 \wedge \cdots \wedge w_k$ means apply ∂_{w_1}, then ∂_{w_2}, and so on, up to ∂_{w_k}. We extend this by linearity to any linear combination of such constructions. The result is well-defined.

2.8.3 General case

With the formalism of alternating algebra in hand, we turn now to formalizing and generalizing the representation that we introduced in Subsection 2.8.1, henceforth the *simplicial representation*. The parameters $k, l \in \mathbb{N}$ may be arbitrary; however, we will continue to restrict to polynomials of degree $d = l + 1$. As before, since $d = l + 1$, polynomials of degree d that are not multi-linear are automatically outside $\mathrm{Van}[\mathrm{RFE}_l^k]$, so we also restrict our attention to multi-linear polynomials.

Let $\{x_1, \ldots, x_n\}$ be the set of variables, and correspond each variable x_i with a fresh vertex, labeled i. Degree-$(l+1)$ multi-linear polynomials are represented by linear combinations of $(l+1)$-vertex simplices, and we will use simplices of other dimensions to study membership in $\mathrm{Van}[\mathrm{RFE}_l^k]$. Together, all live in the following spaces:

Definition 2.45 (Space of oriented simplices). For each $t \in \mathbb{N}$, we let

$$\Sigma^t \doteq \mathrm{span}(i_1 \wedge \cdots \wedge i_t : i_1, \ldots, i_t \in [n]) \tag{2.57}$$

denote the space of linear combinations of t-vertex oriented simplices. ◂

To emphasize, the t in Σ^t counts the number of vertices in the simplices; this is one more than the usual notion of dimension of a simplex.

The representation makes use of certain Vandermonde determinants. We abbreviate them using the following notation:

$$|i_1 \wedge \cdots \wedge i_t|_a \doteq \begin{vmatrix} a_{i_1}^{t-1} & \cdots & 1 \\ \vdots & & \vdots \\ a_{i_t}^{t-1} & \cdots & 1 \end{vmatrix}. \tag{2.58}$$

Note that the determinants depend on the abscissas a_i that underlie RFE.

Algebraically, the representation of a polynomial in this representation can be concisely understood as follows. For distinct $i_1, \ldots, i_{l+1} \in [n]$, the monomial $x_{i_1} \cdots x_{i_{l+1}}$ is represented as the following element of Σ^{l+1}:

$$\frac{i_1 \wedge \cdots \wedge i_{l+1}}{|i_1 \wedge \cdots \wedge i_{l+1}|_a}. \tag{2.59}$$

Note that the above is indeed symmetric in the order of the variable indices: exchanging any two indices causes both the numerator and denominator to change signs, to a net effect of no change. Formally, we define the following "decoder map" that maps a simplicial representation to the polynomial it represents:

Definition 2.46 (Representation). Let $\rho : \Sigma^{l+1} \to \mathbb{F}[x_1, \ldots, x_n]$ be the linear map extending

$$i_1 \wedge \cdots \wedge i_{l+1} \;\mapsto\; |i_1 \wedge \cdots \wedge i_{l+1}|_a \cdot x_{i_1} \cdots x_{i_{l+1}}. \tag{2.60}$$

◂

ρ is a vector space isomorphism between Σ^{l+1} and the space of homogeneous degree-$(l+1)$ multi-linear polynomials.

Reasoning about membership in Van[RFE] makes use of boundary operators $\bigoplus_{t=0}^{n} \Sigma^t \to \bigoplus_{t=0}^{n} \Sigma^t$. Boundary operators are parametrized by functions $w : [n] \to \mathbb{F}$. For our purposes, it is useful to view these functions as *univariate polynomials* in the sense that a univariate polynomial $w \in \mathbb{F}[\alpha]$ determines the function $i \mapsto w(a_i)$.

Definition 2.47 (Boundary operator). For any univariate polynomial $w \in \mathbb{F}[\alpha]$, the boundary operator with weight function w, $\partial_w : \bigoplus_{t=0}^{n} \Sigma^t \to \bigoplus_{t=0}^{n} \Sigma^t$, is defined to be the linear map extending

$$i_1 \wedge \cdots \wedge i_t \;\mapsto\; \sum_{j=1}^{t} (-1)^{1+j} w(a_{i_j}) \; (i_1 \wedge \cdots \wedge i_{j-1} \wedge i_{j+1} \wedge \cdots \wedge i_t). \tag{2.61}$$

◂

For each $t \geq 1$, $\partial_w(\Sigma^t) \subseteq \Sigma^{t-1}$, while $\partial_w(\Sigma^0) = \{0\}$.

One may restrict attention to w with degree less than n, since any two polynomials that are the same as functions $\{a_i : i \in [n]\} \to \mathbb{F}$ determine the same boundary operator. In general, we will be interested in the boundaries that are weighted by low-degree polynomials. When w has degree δ or less, we say that ∂_w is a degree-δ boundary.

As discussed in Subsection 2.8.2, boundary operators under composition behave like an alternating algebra. In this way, the expression ∂_ω for some $\omega = w_1 \wedge \cdots \wedge w_r$ is defined to be $\partial_{w_r} \circ \cdots \circ \partial_{w_1}$.

We can now describe the simplicial representation of EVC:

Lemma 2.48. *For any $k, l \in \mathbb{N}$ and $i_1, \ldots, i_{k+l+2} \in [n]$,*

$$\mathrm{EVC}_l^k[i_1, \ldots, i_{k+l+2}] \;=\; (-1)^{(k+1)(l+1)} \cdot \rho\left(\partial_{\alpha^k \wedge \ldots \wedge \alpha^0} \left(i_1 \wedge \cdots \wedge i_{k+l+2} \right) \right). \tag{2.62}$$

That is, EVC_l^k is the polynomial formed (up to sign) from a given $(l+1)$-vertex simplex by iteratively applying to it the $k+1$ boundaries weighted by $\alpha^0, \alpha^1, \ldots, \alpha^k$. The sign factor in (2.62) can be removed if one rearranges (2.4) so that the first $l+1$ columns come after the last $k+1$ columns.

Every degree-$(l+1)$ polynomial in $\mathrm{Van}[\mathrm{RFE}_l^k]$ is a linear combination of instantiations of EVC_l^k, which is to say that it is in the image of Σ^{k+l+2} through $\rho \circ \partial_{\alpha^k \wedge \ldots \wedge \alpha^0}$. Equivalently, for any degree-$(l+1)$ homogeneous multi-linear polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$, p belongs to $\mathrm{Van}[\mathrm{RFE}_l^k]$ if and only if $\rho^{-1}(p)$ is in the set $\partial_{\alpha^k \wedge \ldots \wedge \alpha^0}(\Sigma^{l+k+2})$. The following relationship is an instance of a general phenomenon in alternating algebra:

$$\mathrm{Im}\left(\partial_{\alpha^k \wedge \ldots \wedge \alpha^0} \right) \;=\; \bigcap_{r=0}^{k} \mathrm{Ker}\left(\partial_{\alpha^r} \right). \tag{2.63}$$

This leads to the following characterization of the degree-$(l+1)$ elements of $\mathrm{Van}[\mathrm{RFE}_l^k]$:

Theorem 2.49. *Let $k, l \in \mathbb{N}$. For any homogeneous multi-linear polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ of degree $l+1$, $p(\mathrm{RFE}_l^k) = 0$ if and only if*

$$\partial_w(\ \rho^{-1}(p)\ ) = 0 \tag{2.64}$$

for every $w \in \mathbb{F}[\alpha]$ of degree at most k.

In other words, for any degree-$(l+1)$ homogeneous multi-linear polynomial, it vanishes at RFE_l^k if and only if in the simplicial representation it satisfies conservation with respect to all degree-k boundaries. This is the representation and ideal membership characterization for such polynomials for general k and l.

Theorem 2.49 is ultimately a reformulation of Theorem 2.4 (more precisely, Lemma 2.18) in the specific case of homogeneous, multi-linear polynomials of degree $l+1$. For sets K and L as in Lemma 2.18, let $w(\alpha) \doteq \prod_{y \in K}(\alpha - a_y)$ and let $i_1, \ldots, i_l$ enumerate L. The coefficient of $i_1 \wedge \cdots \wedge i_l$ in $\partial_w(\rho^{-1}(p))$, multiplied by $|i_1 \wedge \cdots \wedge i_l|_a$, precisely equals the evaluation of $\left(\frac{\partial p}{\partial K}\right)\Big|_{L \leftarrow 0}$ at (2.32).

Theorem 2.49 extends to higher degrees as follows. Let $d > l$, and let $\mathbb{F}[x_1, \ldots, x_n]_{=d}$ be the space of degree-d homogeneous polynomials. For each monomial $x_{i_1} \cdots x_{i_d} \in \mathbb{F}[x_1, \ldots, x_n]_{=d}$ with distinct $i_1, \ldots, i_d$, we represent it as $i_1 \wedge \cdots \wedge i_d / |i_1 \wedge \cdots \wedge i_d|_a$. This means extending our decoder map ρ to Σ^d:

$$\begin{array}{rccc} \rho & : & \Sigma^d & \to & \mathbb{F}[x_1, \ldots, x_n]_{=d} \\ & & i_1 \wedge \cdots \wedge i_d & \mapsto & |i_1 \wedge \cdots \wedge i_d|_a \cdot x_{i_1} \cdots x_{i_d} \end{array} . \tag{2.65}$$

ρ is a vector space isomorphism between Σ^d and the multi-linear subspace of $\mathbb{F}[x_1, \ldots, x_n]_{=d}$.

The following theorem characterizes $\mathrm{Van}[\mathrm{RFE}_l^k]$ within this representation. It is likewise ultimately a reformulation of Theorem 2.4 (or Lemma 2.18 to be precise).

Theorem 2.50. *Let $k, l \in \mathbb{N}$ and $\Delta \geq 1$. For any homogeneous multi-linear polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ of degree $l + \Delta$, $p(\mathrm{RFE}_l^k) = 0$ if and only if*

$$\partial_{w_1 \wedge \cdots \wedge w_\Delta}(\rho^{-1}(p)) = 0 \tag{2.66}$$

for every $w_1, \ldots, w_\Delta \in \mathbb{F}[\alpha]$ of degree at most $k + \Delta - 1$.

Theorems 2.49 and 2.50 do well for understanding the multi-linear elements of the vanishing ideal. For non-multi-linear elements, one may do the following. Let $\widehat{\Sigma}^t$ be Σ^t except that coefficients may be arbitrary polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ rather than just scalars in $\mathbb{F}$. The decoder map ρ and boundary maps ∂_w carry over to $\widehat{\Sigma}^t$ directly, though now ρ is no longer injective. The following variation of Theorem 2.49 characterizes ideal membership for arbitrary polynomials.

Theorem 2.51. *Let $k, l \in \mathbb{N}$. For any polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$, $p(\mathrm{RFE}_l^k) = 0$ if and only if there exists $\eta \in \widehat{\Sigma}^{l+1}$ with $\rho(\eta) = p$ such that, for every $w \in \mathbb{F}[\alpha]$ of degree at most k,*

$$\partial_w(\eta) = 0. \tag{2.67}$$

While Theorem 2.51 applies to a broader class of polynomials, it has the drawback that representing polynomials with $\widehat{\Sigma}^{l+1}$ is too redundant. Specifically, whenever p has a representation in $\widehat{\Sigma}^{l+1}$, there are many $\eta \in \widehat{\Sigma}^{l+1}$ that represent p, and most of them do *not* satisfy the boundary conditions, even when p belongs to the vanishing ideal. This weakens the utility of the characterization. Theorems 2.49 and 2.50 yield straightforward tests: given p, form the unique η with $\rho(\eta) = p$, and then check whether the boundary conditions hold for η. Theorem 2.51, on the other hand, leaves η comparatively underspecified.

Chapter 3

Circuit Minimization

3.1 Overview

Finding a circuit of minimum size that computes a given Boolean function constitutes the overarching goal in nonuniform complexity theory. It defines an interesting computational problem in its own right, the complexity of which depends on the way the Boolean function is specified. A generic and natural, albeit verbose, way to specify a Boolean function is via its truth table. The corresponding decision problem is known as the Minimum Circuit Size Problem (MCSP): Given a truth table and a threshold θ, does there exist a Boolean circuit of size at most θ that computes the Boolean function specified by the truth table? The interest in MCSP dates back to the dawn of theoretical computer science [Tra84]. It continues today partly due to the fundamental nature of the problem, and partly because of the work on natural proofs and the connections between pseudorandomness and computational hardness.

A closely related problem from Kolmogorov complexity theory is the Minimum KT Problem (MKTP), which deals with compression in the form of efficient programs instead of circuits. Rather than asking if the input has a small circuit when interpreted as the truth table of a Boolean function, MKTP asks if the input has a small program that produces each individual bit of the input quickly. To be more specific, let us fix a universal Turing

machine U. We consider descriptions of the input string x in the form of a program d such that, for every bit position i, U on input d and i outputs the i-th bit of x in T steps. The KT cost of such a description is defined as $|d| + T$, i.e., the bit-length of the program plus the running time. The KT complexity of x, denoted $\mathrm{KT}(x)$, is the minimum KT cost of a description of x. $\mathrm{KT}(x)$ is polynomially related to the circuit complexity of x when viewed as a truth table (see Section 3.2.1 for a more formal treatment). On input a string x and an integer θ, MKTP asks whether $\mathrm{KT}(x) \leq \theta$.

Both MCSP and MKTP are in NP but are not known to be in P or NP-complete. As such, they are two prominent candidates for NP-intermediate status. Others include factoring integers, discrete log over prime fields, graph isomorphism (GI), and a number of similar isomorphism problems.

Whereas NP-complete problems all reduce one to another, even under fairly simple reductions, less is known about the relative difficulty of presumed NP-intermediate problems. Regarding MCSP and MKTP, despite their apparent similarity, it is not known whether one reduces to the other. Factoring integers and discrete log over prime fields are known to reduce to both MCSP and MKTP under randomized reductions with zero-sided error [ABK+06; Rud17]. Recently, Allender and Das [AD17] showed that GI and all of SZK (Statistical Zero Knowledge) reduce to both under randomized reductions with bounded error.

Those reductions and, prior to this work, all reductions of supposedly-intractable problems to MCSP / MKTP, proceed along the same well-trodden path. Namely, MCSP / MKTP is used as an efficient statistical test to distinguish random distributions from pseudorandom distributions, where the pseudorandom distribution arises from a hardness-based pseudorandom generator construction. In particular, [KC00] employs the construction based on the hardness of factoring Blum integers, [ABK+06; AD17; AKR+10; Rud17] use the construction from [HIL+99] based on the existence of one-way functions, and [ABK+06; CIK+16] make use of the Nisan-Wigderson construction [NW94]. The property that MCSP / MKTP breaks the construction implies that the underlying hardness assumption fails relative to

MCSP / MKTP, and thus that the supposedly hard problem reduces to MCSP / MKTP.

Contributions The main conceptual contribution in this chapter is a fundamentally different way of constructing reductions to MKTP based on a novel use of known interactive proof systems. The approach applies to GI and a broad class of isomorphism problems. A common framework for those isomorphism problems is another conceptual contribution. In terms of results, the new approach allows us to eliminate the errors in the known reductions from GI to MKTP, and more generally to establish *zero-sided error* randomized reductions to MKTP from many isomorphism problems within the framework. These include Linear Code Equivalence, Matrix Subspace Conjugacy, and Permutation Group Conjugacy (see Section 3.6 for the definitions). Many other isomorphism problems reduce to Matrix Subspace Conjugacy [GQ21], so the result extends automatically to all those problems as well. The technical contributions mainly consist of encodings of isomorphism classes that are efficiently decodable and achieve compression that is at or near the information-theoretic optimum.

The techniques remain of interest even in light of the quasi-polynomial-time algorithm for GI [Bab16]. For one, GI is still not known to be in P, and Group Isomorphism stands as a significant obstacle to this (as stated at the end of [Bab16]). More importantly, the techniques also apply to the other isomorphism problems mentioned above, for which the current best algorithms are still exponential.

In addition, there is some evidence that our approach for constructing reductions to MKTP differs in an important way from the existing ones. Specifically, the existing approach can only yield zero-sided error reductions to MKTP from problems that are in $\mathrm{NP} \cap \mathrm{coNP}$, a class that neither GI nor any of the other isomorphism problems mentioned above are known to reside in. This is because the underlying hardness assumptions are fundamentally average-case,[1] which implies that the reduction can have both false positives and false negatives. For example, in the papers employing the construction from [HIL+99], MKTP

[1] In some settings worst-case to average-case reductions are known, but these reductions are themselves randomized with two-sided error.

is used in a subroutine to invert a polynomial-time-computable function (see Lemma 3.3 in Section 3.2.1), and the subroutine may fail to find an inverse. Given a reliable but imperfect subroutine, the traditional way to eliminate false positives is to use the subroutine for constructing an efficiently verifiable membership witness, and only accept after verifying its validity. As such, the existence of a traditional reduction without false positives from a language L to MKTP implies that $L \in \mathrm{NP}$. Similarly, a traditional reduction from L to MKTP without false negatives is only possible if $L \in \mathrm{coNP}$, and zero-sided error is only possible if $L \in \mathrm{NP} \cap \mathrm{coNP}$.

Main Idea Instead of using the oracle for MKTP in the *construction* of a candidate witness and then verifying the validity of the candidate without the oracle, the oracle is used in the *verification* process. This obviates the need for the language L to be in $\mathrm{NP} \cap \mathrm{coNP}$ in the case of reductions with zero-sided error.

Let us see how to implement this idea for $L = \mathrm{GI}$. Recall that an instance of GI consists of a pair (G_0, G_1) of graphs on the vertex set $[n]$, and the question is whether $G_0 \equiv G_1$, i.e., whether there exists a permutation $\pi \in S_n$ such that $G_1 = \pi(G_0)$, where $\pi(G_0)$ denotes the result of applying the permutation π to the vertices of G_0. In order to develop a zero-sided error algorithm for GI, it suffices to develop one without false negatives. This is because the false positives can subsequently be eliminated using the known search-to-decision reduction for GI [KST93].

The crux for obtaining a reduction without false negatives from GI to MKTP is a witness system for the complement $\overline{\mathrm{GI}}$ inspired by the well-known two-round interactive proof system for $\overline{\mathrm{GI}}$ [GMW91]. Consider the distribution $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random. By the Orbit–Stabilizer Theorem, for any fixed G, R_G is uniform over a set of size $N \doteq n!/|\mathrm{Aut}(G)|$ and thus has entropy $s = \log(N)$, where $\mathrm{Aut}(G) \doteq \{\pi \in S_n : \pi(G) = G\}$ denotes the set of automorphisms of G. For ease of exposition, let us assume that $|\mathrm{Aut}(G_0)| = |\mathrm{Aut}(G_1)|$ (which is actually the hardest case for GI), so both R_{G_0} and R_{G_1} have

the same entropy s. Consider picking $r \in \{0,1\}$ uniformly at random, and setting $G = G_r$. If $(G_0, G_1) \in \mathrm{GI}$, the distributions R_{G_0}, R_{G_1}, and R_G are all identical, and therefore R_G also has entropy s. On the other hand, if $(G_0, G_1) \notin \mathrm{GI}$, the entropy of R_G equals $s+1$. The extra bit of information corresponds to the fact that in the nonisomorphic case each sample of R_G reveals the value of r that was used, whereas that bit gets lost in the reduction in the isomorphic case.

The difference in entropy suggests that a typical sample of R_G can be compressed more in the isomorphic case than in the nonisomorphic case. If we can compute some threshold such that $\mathrm{KT}(R_G)$ *never* exceeds the threshold in the isomorphic case, and exceeds it with non-negligible probability in the nonisomorphic case, we have the witness system for $\overline{\mathrm{GI}}$ that we aimed for: Take a sample from R_G, and use the oracle for MKTP to check that it cannot be compressed at or below the threshold. The entropy difference of 1 may be too small to discern, but we can amplify the difference by taking multiple samples and concatenating them. Thus, we end up with a randomized mapping reduction of the following form, where t denotes the number of samples and θ the threshold:

$$\begin{aligned} &\text{Pick } r \doteq r_1 \dots r_t \in \{0,1\}^t \text{ and } \pi_i \in S_n \text{ for } i \in [t], \text{ uniformly at random.} \\ &\text{Output } (y, \theta) \text{ where } y \doteq y_1 \dots y_t \text{ and } y_i \doteq \pi_i(G_{r_i}). \end{aligned} \tag{3.1}$$

We need to analyze how to set the threshold θ and argue correctness for a value of t that is polynomially bounded. In order to do so, let us first consider the case where the graphs G_0 and G_1 are *rigid*, i.e., they have no nontrivial automorphisms, or equivalently, $s = \log(n!)$.

- If $G_0 \not\equiv G_1$, the string y contains all of the information about the random string r and the t random permutations $\pi_1, \dots, \pi_t$, which amounts to $ts + t = t(s+1)$ bits of information. This implies that y has KT-complexity close to $t(s+1)$ with high probability.

- If $G_0 \equiv G_1$, then we can efficiently produce each bit of y from the adjacency matrix representation of G_0 (n^2 bits) and the function table of permutations $\tau_i \in S_n$ (for $i \in [t]$)

such that $y_i \doteq \pi_i(G_{r_i}) = \tau_i(G_0)$. Moreover, the set of all permutations S_n allows an efficiently decodable indexing, i.e., there exists an efficient algorithm that takes an index $k \in [n!]$ and outputs the function table of the k-th permutation in S_n according to some ordering. An example of such an indexing is the Lehmer code (see, e.g., [Knu98, pp. 12-13] for specifics). This shows that

$$\mathrm{KT}(y) \leq t\lceil s \rceil + (n + \log(t))^c \tag{3.2}$$

for some constant c, where the first term represents the cost of the t indices of $\lceil s \rceil$ bits each, and the second term represents the cost of the n^2 bits for the adjacency matrix of G_0 and the polynomial running time of the decoding process.

If we ignore the difference between s and $\lceil s \rceil$, the right-hand side of (3.2) becomes $ts + n^c$, which is closer to ts than to $t(s+1)$ for t any sufficiently large polynomial in n, say $t = n^{c+1}$. Thus, setting θ halfway between ts and $t(s+1)$, i.e., $\theta \doteq t(s + \frac{1}{2})$, ensures that $\mathrm{KT}(y) > \theta$ holds with high probability if $G_0 \not\equiv G_1$, and never holds if $G_0 \equiv G_1$. This yields the desired randomized mapping reduction without false negatives, modulo the rounding issue of s to $\lceil s \rceil$. The latter can be handled by aggregating the permutations τ_i into blocks so as to make the amortized cost of rounding negligible. The details are captured in the Blocking Lemma of Section 3.3.1.

What changes in the case of non-rigid graphs? For ease of exposition, let us again assume that $|\mathrm{Aut}(G_0)| = |\mathrm{Aut}(G_1)|$. There are two complications:

(i) We no longer know how to efficiently compute the threshold $\theta \doteq t(s + \frac{1}{2})$ because $s \doteq \log(N)$ where $N \doteq n!/|\mathrm{Aut}(G_0)| = n!/|\mathrm{Aut}(G_1)|$ involves the size of the automorphism group.

(ii) The Lehmer code no longer provides sufficient compression in the isomorphic case as it requires $\log(n!)$ bits per permutation whereas we only have s to spend, which could be considerably less than $\log(n!)$.

In order to resolve (ii) we develop an efficiently decodable indexing of cosets for any subgroup of S_n given by a list of generators (see Lemma 3.9 in Section 3.3.2). In fact, our scheme even works for cosets of a subgroup within another subgroup of S_n (see Lemma 3.30 in Section 3.7). Applying our scheme to $\mathrm{Aut}(G)$ and including a minimal list of generators for $\mathrm{Aut}(G)$ in the description of the program p allows us to maintain (3.2).

Regarding (i), we can deduce a good approximation to the threshold with high probability by taking, for both choices of $r \in \{0,1\}$, a polynomial number of samples of R_{G_r} and using the oracle for MKTP to compute the exact KT-complexity of their concatenation. This leads to a randomized reduction from GI to MKTP with bounded error (from which one without false positives follows as mentioned before), reproving the earlier result of [AD17] using our new approach (see Remark 3.10 in Section 3.3.2 for more details).

In order to avoid false negatives, we need to improve the above approximation algorithm such that it never produces a value that is too small, while maintaining efficiency and the property that it outputs a good approximation with high probability. In order to do so, it suffices to develop a *probably-correct overestimator* for the quantity $n!/|\mathrm{Aut}(G)|$, i.e., a randomized algorithm that takes as input an n-vertex graph G, produces the correct quantity with high probability, and never produces a value that is too small; the algorithm should run in polynomial time with access to an oracle for MKTP. Equivalently, it suffices to develop a probably-correct *under*estimator of similar complexity for $|\mathrm{Aut}(G)|$.

The latter can be obtained from the known search-to-decision procedures for GI, and answering the oracle calls to GI using the above two-sided error reduction from GI to MKTP. There are a number of ways to implement this strategy; here is one that generalizes to a number of other isomorphism problems including Linear Code Equivalence.

1. Find a list of permutations that generates a subgroup of $\mathrm{Aut}(G)$ such that the subgroup equals $\mathrm{Aut}(G)$ with high probability.

   Finding a list of generators for $\mathrm{Aut}(G)$ deterministically reduces to GI (see, e.g., sec-

tions 1.2–1.3 in [KST93]).[2] Substituting the oracle for GI by a two-sided error algorithm yields a list of permutations that generates $\mathrm{Aut}(G)$ with high probability, and always produces a subgroup of $\mathrm{Aut}(G)$. The latter property follows from the inner workings of the reduction, or can be imposed explicitly by checking every permutation produced and dropping it if it does not map G to itself. We use the above randomized reduction from GI to MKTP as the two-sided error algorithm for GI.

2. Compute the order of the subgroup generated by those permutations.

   There is a known deterministic polynomial-time algorithm to do this (see, e.g., [Ser03]).

The resulting probably-correct underestimator for $|\mathrm{Aut}(G)|$ runs in polynomial time with access to an oracle for MKTP. Plugging it into our approach, we obtain a randomized reduction from GI to MKTP without false negatives. A reduction with zero-sided error follows as discussed earlier.

Before applying our approach to other isomorphism problems, let us point out the important role that the Orbit–Stabilizer Theorem plays. A randomized algorithm for finding generators for a graph's automorphism group yields a probably-correct underestimator for the size of the automorphism group, as well as a randomized algorithm for GI without false positives. The Orbit–Stabilizer Theorem allows us to turn a probably-correct underestimator for $|\mathrm{Aut}(G)|$ into a probably-correct overestimator for the size of the support of R_G, thereby switching the error from one side to the other, and allowing us to avoid false negatives instead of false positives.

General Framework The above approach extends to several other isomorphism problems. They can be cast in the following common framework, parametrized by an underlying

[2]Briefly, suppose the vertices of G are $1, 2, \ldots, n$. By appropriately coloring vertices of two separate copies of G, one may query, for each $u = 1, 2, \ldots, n$ and each $v = u + 1, u + 2, \ldots, n$, whether there is an automorphism of G that fixes $1, 2, \ldots, u - 1$ and sends u to v. Moreover, whenever such an automorphism exists, it may be constructed efficiently through further refinement of the colors and use of oracle queries to GI, as in the standard search-to-decision reduction for GI. Such a collection of automorphisms generates $\mathrm{Aut}(G)$.

family of group actions (Ω, H) where H is a group that acts on the universe Ω. We typically think of the elements of Ω as abstract objects, which need to be described in string format in order to be input to a computer; we let $\omega(z)$ denote the abstract object represented by the string z.

Definition 3.1 (Isomorphism Problem). An instance of an *isomorphism problem* consists of a pair $x = (x_0, x_1)$ that determines a universe Ω_x and a group H_x that acts on Ω_x such that $\omega_0(x) \doteq \omega(x_0)$ and $\omega_1(x) \doteq \omega(x_1)$ belong to Ω_x. Each $h \in H_x$ is identified with the permutation $h : \Omega_x \to \Omega_x$ induced by the action. The goal is to determine whether there exists $h \in H_x$ such that $h(\omega_0(x)) = \omega_1(x)$. ◂

When it causes no confusion, we drop the argument x and simply write H, Ω, ω_0, and ω_1. We often blur the—sometimes pedantic—distinction between z and $\omega(z)$. For example, in GI, each z is an $n \times n$ binary matrix (a string of length n^2), and represents the abstract object $\omega(z)$ of a graph with n labeled vertices; thus, in this case the correspondence between z and $\omega(z)$ is a bijection. The group H is the symmetric group S_n, and the action is by permuting the labels.

Table 3.1 summarizes how the problems we mentioned earlier can be cast in the framework (see Section 3.6 for details about the last three).

Problem	H	Ω
Graph Isomorphism	S_n	graphs with n labeled vertices
Linear Code Equivalence	S_n	linear subspaces of $\mathbb{F}_q^n$
Permutation Group Conjugacy	S_n	subgroups of S_n
Matrix Subspace Conjugacy	$\mathrm{GL}_n(\mathbb{F}_q)$	linear subspaces of $\mathbb{F}_q^{n\times n}$

Table 3.1: Example isomorphism problems

We generalize our construction for GI to any isomorphism problem by replacing $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random, by $R_\omega(h) \doteq h(\omega)$ where $h \in H$ is chosen uniformly at random. The analysis that the construction yields a randomized reduction without false negatives from the isomorphism problem to MKTP carries over, provided that the isomorphism problem satisfies the following properties.

1. The underlying group H is *efficiently samplable*, and the action $(\omega, h) \mapsto h(\omega)$ is efficiently computable. We need this property in order to make sure the reduction is efficient.

2. There is an efficiently computable *normal form* for representing elements of Ω as strings. This property trivially holds in the setting of GI as there is a unique adjacency matrix that represents any given graph on the vertex set $[n]$. However, uniqueness of representation need not hold in general. Consider, for example, Permutation Group Conjugacy. An instance of this problem abstractly consists of two permutation groups (Γ_0, Γ_1), represented (as usual) by a sequence of elements of S_n generating each group. In that case there are many strings representing the same abstract object, i.e., a subgroup has many different sets of generators.

   For the correctness analysis in the isomorphic case it is important that H acts on the abstract objects, and *not* on the binary strings that represent them. In particular, the output of the reduction should only depend on the abstract object $h(\omega)$, and not on the way ω was provided as input. This is because the latter may leak information about the value of the bit r that was picked. The desired independence can be guaranteed by applying a normal form to the representation before outputting the result. In the case of Permutation Group Conjugacy, this means transforming a set of permutations that generate a subgroup Γ into a canonical set of generators for Γ.

   In fact, it suffices to have an efficiently computable *complete invariant* for Ω, i.e., a mapping from representations of objects from Ω to strings such that the image only depends on the abstract object, and is different for different abstract objects.

3. There exists a probably-correct overestimator for $N \doteq |H|/|\mathrm{Aut}(\omega)|$ that is computable efficiently with access to an oracle for MKTP. We need this property to set the threshold $\theta \doteq t(s + \frac{1}{2})$ with $s \doteq \log(N)$ correctly.

4. There exists an encoding for cosets of $\mathrm{Aut}(\omega)$ in H that achieves KT-complexity close

to the information-theoretic optimum (see Section 3.2.2 for the definition of an encoding). This property ensures that in the isomorphic case the KT-complexity is never much larger than the entropy.

Properties 1 and 2 are fairly basic. Property 4 may seem to require an instantiation-dependent approach. However, in Section 3.4 we shall see a *generic* hashing-based encoding scheme that meets the requirements. In fact, the approach provides a nearly-optimal encoding scheme for any samplable distribution that is almost flat, without reference to isomorphism. Unlike the indexings from Lemma 3.9, the generic construction does not achieve the information-theoretic optimum, but it comes sufficiently close for our purposes.

The notion of a probably-correct overestimator in Property 3 can be further relaxed to that of a *probably-approximately-correct overestimator*, or *pac overestimator* for short. This is a randomized algorithm that with high probability outputs a value within an absolute deviation bound of Δ from the correct value, and never produces a value that is more than Δ below the correct value. More precisely, it suffices to efficiently compute with access to an oracle for MKTP a pac overestimator for $s \doteq \log(|H|/|\mathrm{Aut}(\omega)|)$ with deviation $\Delta = 1/4$. The relaxation suffices because of the difference of about $1/2$ between the threshold θ and the actual KT-values in both the isomorphic and the nonisomorphic case. As $s = \log|H| - \log|\mathrm{Aut}(\omega)|$, it suffices to have a pac overestimator for $\log|H|$ and a pac *under*estimator for $\log|\mathrm{Aut}(\omega)|$, both to within deviation $\Delta/2 = 1/8$ and of the required efficiency.

Generalizing our approach for GI, one way to obtain the desired underestimator for $\log|\mathrm{Aut}(\omega)|$ is by showing how to efficiently compute with access to an oracle for MKTP:

(a) a list L of elements of H that generates a subgroup $\langle L\rangle$ of $\mathrm{Aut}(\omega)$ such that $\langle L\rangle = \mathrm{Aut}(\omega)$ with high probability, and

(b) a pac underestimator for $\log|\langle L\rangle|$, the logarithm of the order of the subgroup generated by a given list L of elements of H.

Following the above approach for GI, we know how to achieve (a) when the isomorphism problem allows a search-to-decision reduction. Such a reduction is known for Linear Code Equivalence, but remains open for problems like Matrix Subspace Conjugacy and Permutation Group Conjugacy. However, we shall see that (a) holds for a *generic* isomorphism problem provided that products and inverses in H can be computed efficiently (see Lemma 3.20 in Section 3.5.2). The proof hinges on the ability of MKTP to break the pseudo-random generator construction of [HIL+99] based on a purported one-way function (Lemma 3.3 in Section 3.2.1).

As for (b), we know how to efficiently compute the order of the subgroup exactly in the case of permutation groups ($H = S_n$), even without an oracle for MKTP, and in some other cases with the MKTP oracle (see, e.g., [BBS09] and subsequent work for matrix groups), though not for all groups. Instead, we show how to *generically* construct a *pac underestimator* with small deviation given access to MKTP as long as products and inverses in H can be computed efficiently, and H allows an efficient complete invariant (see Lemma 3.21 in Section 3.5.2). The first two conditions are sufficient to efficiently generate a distribution $\widetilde{p}$ on $\langle L \rangle$ that is uniform to within a small relative deviation [Bab91]. The entropy $\widetilde{s}$ of that distribution equals $\log|\langle L \rangle|$ to within a small additive deviation. As $\widetilde{p}$ is almost flat, our encoding scheme from Section 3.4 shows that $\widetilde{p}$ has an encoding whose length does not exceed $\widetilde{s}$ by much, and that can be decoded by small circuits. Given an efficient complete invariant for H, the approach we used to pac underestimate the threshold θ carries over to a pac underestimator for $\widetilde{s}$ with small additive deviation, namely the amortized KT-complexity of the concatenation of a polynomial number of samples from $\widetilde{p}$. With access to an oracle for MKTP we can efficiently evaluate KT. As a result, we obtain a pac underestimator for $\log|\langle L \rangle|$ with a small additive deviation that is efficiently computable with oracle access to MKTP.

The above ingredients allow us to conclude that all of the isomorphism problems in Table 3.1 reduce to MKTP under randomized reductions without false negatives. Moreover,

we argue that Properties 1 and 2 are sufficient to generalize the construction of Allender and Das [AD17], which yields randomized reductions of the isomorphism problem to MKTP without false positives (irrespective of whether a search-to-decision reduction is known). By combining both reductions, we conclude that all of the isomorphism problems in Table 3.1 reduce to MKTP under randomized reductions with zero-sided error. See Sections 3.5 and 3.6 for more details.

3.2 Preliminaries

The standard complexity classes BPP, RP, coRP, and ZPP each comprise the problems computable by randomized algorithms in expected polynomial time, and differ only in how the error is bounded. In BPP false positives and false negatives may occur; in RP there can be no false positives; in coRP there can be no false negatives; and in ZPP there can be neither false positives nor false negatives. We make use of these classes and their relativizations that use MKTP as an oracle. For formal definitions, we refer to [AB09].

The remainder of this section provides more details about KT-complexity, formally defines the related notions of indexing and encoding, and reviews some background on graph isomorphism.

3.2.1 KT Complexity

The measure KT that we informally described in Section 3.1, was introduced and formally defined as follows in [ABK+06]. We refer to that paper for more background and motivation for the particular definition.

Definition 3.2 (KT). Let U be a universal Turing machine. For each string x, define $\mathrm{KT}_U(x)$ to be

$$\min\{\, |d| + T : \ (\forall \sigma \in \{0,1,*\})\ (\forall i \leq |x|+1)\ U^d(i,\sigma) \text{ accepts in } T \text{ steps iff } x_i = \sigma \,\}. \quad (3.3)$$

We define $x_i = *$ if $i > |x|$; thus, for $i = |x| + 1$ the machine accepts iff $\sigma = *$. The notation U^d indicates that the machine U has random access to the description d. ◂

$\mathrm{KT}(x)$ is defined to be equal to $\mathrm{KT}_U(x)$ for a fixed choice of universal machine U with logarithmic simulation time overhead [ABK+06, Proposition 5]. In particular, if d consists of the description of a Turing machine M that runs in time $T_M(n)$ and some auxiliary information a such that $M^a(i) = x_i$ for $i \in [n]$, then $\mathrm{KT}(x) \leq |a| + c_M T_M(\log n) \log(T_M(\log n))$, where $n \doteq |x|$ and c_M is a constant depending on M. It follows that $(\mu/\log n)^{\Omega(1)} \leq \mathrm{KT}(x) \leq (\mu \cdot \log n)^{O(1)}$ where μ represents the circuit complexity of the mapping $i \mapsto x_i$ [ABK+06, Theorem 11].

The Minimum KT Problem is defined as $\mathrm{MKTP} \doteq \{(x, \theta) \mid \mathrm{KT}(x) \leq \theta\}$. [ABK+06] showed that an oracle for MKTP is sufficient to invert on average any function that can be computed efficiently. The following is a precise formulation:

Lemma 3.3 (follows from Theorem 45 in [ABK+06]). *There exists a polynomial-time probabilistic Turing machine M using oracle access to* MKTP *so that the following holds. For any circuit C on n input bits,*

$$\Pr[C(\tau) = C(\sigma)] \geq 1/\mathrm{poly}(n) \text{ where } \tau \doteq M(C, C(\sigma)), \tag{3.4}$$

and the probability is over the uniform distribution of $\sigma \in \{0,1\}^n$ and the internal coin flips of M.

3.2.2 Random Variables, Samplers, Indexings and Encodings

A finite probability space consists of a finite sample space S, and a probability distribution p on S. Typical sample spaces include finite groups and finite sets of strings. The probability distributions underlying our probability spaces are always uniform.

A *random variable* R is a mapping from the sample space S to a set T, which typically is the universe Ω of a group action, or a set of strings. The random variable R with the

uniform distribution on S induces a distribution p on T. We sometimes use R to denote the induced distribution p as well.

The support of a distribution p on a set T is the set of elements $\tau \in T$ with positive probability $p(\tau)$. A distribution is *flat* if it is uniform on its support. The *entropy* of a distribution p is the expected value of $\log(1/p(\tau))$. The *min-entropy* of p is the largest real s such that $p(\tau) \leq 2^{-s}$ for every τ in the support of p. The *max-entropy* of p is the least real s such that $p(\tau) \geq 2^{-s}$ for every τ in the support of p. For a flat distribution, the min-, max-, and ordinary entropy coincide and equal the logarithm of the size of the support. For two distributions p and q on the same set T, we say that q approximates p within a factor $1+\delta$ if $q(\tau)/(1+\delta) \leq p(\tau) \leq (1+\delta) \cdot q(\tau)$ for all $\tau \in T$. In that case, p and q have the same support, and if p has min-entropy s, then q has min-entropy at least $s - \log(1+\delta)$, and if p has max-entropy s, then q has max-entropy at most $s + \log(1+\delta)$.

A *sampler* within a factor $1+\delta$ for a distribution p on a set T is a random variable $R : \{0,1\}^\ell \to T$ that induces a distribution that approximates p within a factor $1+\delta$. We say that R *samples* T *within a factor* $1+\delta$ *from length* ℓ. If $\delta = 0$ we call the sampler *exact*. The choice of $\{0,1\}^\ell$ reflects the fact that distributions need to be generated from a source of random bits. Factors $1+\delta$ with $\delta > 0$ are necessary in order to sample uniform distributions whose support is not a power of 2.

We consider ensembles of distributions $\{p_x\}$ where x ranges over $\{0,1\}^*$. We call the ensemble *samplable by polynomial-size circuits* if there exists an ensemble of random variables $\{R_{x,\delta}\}$ where δ ranges over the positive rationals such that $R_{x,\delta}$ samples p_x within a factor $1+\delta$ from length $\ell_{x,\delta}$ and $R_{x,\delta}$ can be computed by a circuit of size $\mathrm{poly}(|x|/\delta)$. We stress that the circuits can depend on the string x, not just on $|x|$. If in addition the mappings $(x,\delta) \mapsto \ell_{x,\delta}$ and $(x,\delta,\sigma) \mapsto R_{x,\delta}(\sigma)$ can be computed in time $\mathrm{poly}(|x|/\delta)$, we call the ensemble *uniformly samplable in polynomial time*.

One way to obtain strings with high KT-complexity is as samples from distributions with high min-entropy.

Proposition 3.4. *Let y be sampled from a distribution with min-entropy s. For all k, we have* $\mathrm{KT}(y) \geq \lfloor s-k \rfloor$ *except with probability at most* 2^{-k}.

Proof. There are only $\sum_{i=0}^{\lfloor s-k \rfloor - 1} 2^i < 2^{s-k}$ descriptions of strings with complexity less than $\lfloor s-k \rfloor$. In a distribution with min-entropy s, every sample occurs with probability at most 2^{-s}. Thus the total probability mass on samples with complexity less than $\lfloor s-k \rfloor$ is at most $2^{s-k} \cdot 2^{-s} = 2^{-k}$. ∎

One way to establish upper bounds on KT-complexity is via efficiently decodable encodings into integers from a small range. Encodings with the minimum possible range are referred to as indexings. We use these notions in various settings. The following formal definition is for use with random variables and is general enough to capture all the settings we need. It defines an encoding via its decoder D; the range of the encoding corresponds to the domain of D.

Definition 3.5 (Encoding, Indexing). Let $R : S \to T$ be a random variable. An *encoding* of R is a mapping $D : [N] \to S$ such that for every $\tau \in R(S)$ there exists $i \in [N]$ such that $R(D(i)) = \tau$. We refer to $\lceil \log(N) \rceil$ as the *length* of the encoding. An *indexing* is an encoding with $N = |R(S)|$. ◂

Definition 3.5 applies to a set S by identifying S with the random variable that is the identity mapping on S. It applies to the cosets of a subgroup Γ of a group H by considering the random variable that maps $h \in H$ to the coset $h\Gamma$. It applies to a distribution induced by a random variable R by considering the random variable R itself.

We say that an ensemble of encodings $\{D_x\}$ is *decodable by polynomial-size circuits* if for each x there is a circuit of size $\mathrm{poly}(|x|)$ that computes $D_x(i)$ for every $i \in [N_x]$. If in addition the mapping $(x, i) \mapsto D_x(i)$ is computable in time $\mathrm{poly}(|x|)$, we call the ensemble *uniformly decodable in polynomial time.*

3.2.3 Graph Isomorphism and the Orbit-Stabilizer Theorem

Graph Isomorphism (GI) is the computational problem of deciding whether two graphs, given as input, are isomorphic. A *graph* for us is a simple, undirected graph, that is, a vertex set $V(G)$, and a set $E(G)$ of unordered pairs of vertices. An *isomorphism* between two graphs G_0, G_1 is a bijection $\pi: V(G_0) \to V(G_1)$ that preserves both edges and non-edges: $(v, w) \in E(G_0)$ if and only if $(\pi(v), \pi(w)) \in E(G_1)$. An isomorphism from a graph to itself is an *automorphism*; the automorphisms of a given graph G form a group under composition, denoted $\mathrm{Aut}(G)$. The Orbit–Stabilizer Theorem implies that the number of distinct graphs isomorphic to G equals $n!/|\mathrm{Aut}(G)|$. A graph G is *rigid* if $|\mathrm{Aut}(G)| = 1$, i.e., the only automorphism is the identity, or equivalently, all $n!$ permutations of G yield distinct graphs.

More generally, let H be a group acting on a universe Ω. For $\omega \in \Omega$, each $h \in H$ is an isomorphism from ω to $h(\omega)$. $\mathrm{Aut}(\omega)$ is the set of isomorphisms from ω to itself. By the Orbit–Stabilizer Theorem the number of distinct isomorphic copies of ω equals $|H|/|\mathrm{Aut}(\omega)|$.

3.3 Graph Isomorphism

In this section we show:

Theorem 3.6. $\mathrm{GI} \in \mathrm{ZPP}^{\mathrm{MKTP}}$.

The crux is the randomized mapping reduction from deciding whether a given pair of n-vertex graphs (G_0, G_1) is in GI to deciding whether $(y, \theta) \in \mathrm{MKTP}$, as prescribed by (3.1). Recall that (3.1) involves picking a string $r \doteq r_1 \dots r_t \in \{0,1\}^t$ and permutations π_i at random, and constructing the string $y = y_1 \dots y_t$, where $y_i = \pi_i(G_{r_i})$. It remains to determine θ such that a sufficiently large polynomial t guarantees that the reduction has no false negatives. We follow the outline of Section 3.1, take the same four steps, and fill in the missing details.

3.3.1 Rigid Graphs

We first consider the simplest setting, in which both G_0 and G_1 are rigid. We argue that $\theta \doteq t(s + \frac{1}{2})$ works, where $s = \log(n!)$.

Nonisomorphic Case If $G_0 \not\equiv G_1$, then (by rigidity), each choice of r and each distinct sequence of t permutations results in a different string y, and thus the distribution on the strings y has entropy $t(s+1)$ where $s \doteq \log(n!)$. Thus, by Proposition 3.4, $\mathrm{KT}(y) > \theta = t(s+1) - \frac{t}{2}$ with all but exponentially small probability in t. Thus with high probability the algorithm declares G_0 and G_1 nonisomorphic.

Isomorphic Case If $G_0 \equiv G_1$, we need to show that $\mathrm{KT}(y) \leq \theta$ always holds. The key insight is that the information in y is precisely captured by the t permutations $\tau_1, \tau_2, \ldots, \tau_t$ such that $\tau_i(G_0) = y_i$. These permutations exist because $G_0 \equiv G_1$; they are unique by the rigidity assumption. Thus, y contains at most ts bits of information. We argue that its KT-complexity is not much larger than this. The argument relies on the following encoding, due to Lehmer (see, e.g., [Knu98, pp. 12–33]):

Proposition 3.7 (Lehmer code). *The symmetric groups S_n have indexings that are uniformly decodable in time* $\mathrm{poly}(n)$.

To bound $\mathrm{KT}(y)$, we consider a program d that has the following information hard-wired into it: n, the adjacency matrix of G_0, and the t integers $k_1, \ldots, k_t \in [n!]$ encoding $\tau_1, \ldots, \tau_t$. We use the decoder from Proposition 3.7 to compute the i-th bit of y on input i. This can be done in time $\mathrm{poly}(n, \log(t))$ given the hard-wired information.

As mentioned in Section 3.1, a naïve method for encoding the indices $k_1, \ldots, k_t$ only gives the bound $t\lceil s \rceil + \mathrm{poly}(n, \log(t))$ on $\mathrm{KT}(y)$, which may exceed $t(s+1)$ and—*a fortiori*—the threshold θ, no matter how large a polynomial t is. We remedy this by aggregating multiple indices into blocks, and amortizing the encoding overhead across multiple samples. The following technical lemma captures the technique. For a set T of strings and $b \in \mathbb{N}$, the

statement uses the notation T^b to denote the set of concatenations of b strings from T; see Section 3.2.2 for the other terminology.

Lemma 3.8 (Blocking Lemma). *Let $\{T_x\}$ be an ensemble of sets of strings such that all strings in T_x have the same length* $\mathrm{poly}(|x|)$*. Suppose that for each $x \in \{0,1\}^*$ and $b \in \mathbb{N}$, there is a random variable $R_{x,b}$ whose image contains $(T_x)^b$, and such that $R_{x,b}$ is computable by a circuit of size* $\mathrm{poly}(|x|, b)$ *and has an encoding of length $s'(x,b)$ decodable by a circuit of size* $\mathrm{poly}(|x|, b)$*. Then there are constants c_1 and c_2 so that, for every constant $\alpha > 0$, every $t \in \mathbb{N}$, every sufficiently large x, and every $y \in (T_x)^t$*

$$\mathrm{KT}(y) \le t^{1-\alpha} \cdot s'(x, \lceil t^\alpha \rceil) \; + \; t^{\alpha \cdot c_1} \cdot |x|^{c_2}. \tag{3.5}$$

Let us first see how to apply the Blocking Lemma and then prove it. For a given rigid graph G, we let T_G be the image of the random variable R_G that maps $\pi \in S_n$ to $\pi(G)$ (an adjacency matrix viewed as a string of n^2 bits). We let $R_{G,b}$ be the b-fold Cartesian product of R_G, i.e., $R_{G,b}$ takes in b permutations $\tau_1, \ldots, \tau_b \in S_n$, and maps them to $\tau_1(G)\tau_2(G)\cdots\tau_b(G)$. $R_{G,b}$ is computable by (uniform) circuits of size $\mathrm{poly}(n, b)$. To encode an outcome $\tau_1(G)\tau_2(G)\cdots\tau_b(G)$, we use as index the number whose base-$(n!)$ representation is written $k_1 k_2 \cdots k_b$, where k_i is the index of τ_i from the Lehmer code. This indexing has length $s'(G,b) \doteq \lceil \log(n!^b) \rceil \le bs + 1$. Given an index, the list of permutations $\tau_1, \ldots, \tau_b$ can be decoded by (uniform) circuits of size $\mathrm{poly}(n, b)$. By the Blocking Lemma, we have that

$$\mathrm{KT}(y) \le t^{1-\alpha}(\lceil t^\alpha \rceil s + 1) + t^{\alpha c_1} \cdot n^{c_2} \le ts + t^{1-\alpha} \cdot n^{c_0} + t^{\alpha c_1} \cdot n^{c_2} \tag{3.6}$$

for some constants c_0, c_1, c_2, every constant $\alpha > 0$, and all sufficiently large n, where we use the fact that $s = \log n! \le n^{c_0}$. Setting $\alpha = \alpha_0 \doteq 1/(c_1 + 1)$, this becomes $\mathrm{KT}(y) \le ts + t^{1-\alpha_0} n^{(c_0+c_2)}$. Taking $t = n^{1+(c_0+c_2)/\alpha_0}$, we see that for all sufficiently large n, $\mathrm{KT}(y) \le t(s + \frac{1}{2}) \doteq \theta$.

Proof of the Blocking Lemma. Let $R_{x,b}$ be the hypothesized random variables and $D_{x,b}$ their corresponding decoders. Fix x and t, let $m = \mathrm{poly}(|x|)$ denote the length of the strings in T_x, and let $b \in \mathbb{N}$ be a parameter to be set later.

To bound $\mathrm{KT}(y)$, we first break y into $\lceil t/b \rceil$ blocks $\widetilde{y}_1, \widetilde{y}_2, \ldots, \widetilde{y}_{\lceil t/b \rceil}$ where each $\widetilde{y}_i \in (T_x)^b$ (after padding $\widetilde{y}_{\lceil t/b \rceil}$ with arbitrary strings from T_x if needed). As the image of $R_{x,b}$ contains $(T_x)^b$, each $\widetilde{y}_j$ is encoded by some index k_j of length $s'(x,b)$.

Consider a program d that has x, t, m, b, the circuit for computing $R_{x,b}$, the circuit for computing $D_{x,b}$, and the indices $k_1, k_2, \ldots, k_{\lceil t/b \rceil}$ hardwired, takes an input $i \in \mathbb{N}$, and determines the i-th bit of y as follows. If $i > tm$, then the output is $*$. Otherwise, d first computes $j_0, j_1 \in \mathbb{N}$ so that i points to the j_1-th bit position in $\widetilde{y}_{j_0}$. Then, using $D_{x,b}$, k_{j_0}, and j_1, it finds σ such that $R_{x,b}(\sigma)$ equals $\widetilde{y}_{j_0}$. Finally, it computes $R_{x,b}(\sigma)$ and outputs the j_1-th bit, which is the i-th bit of y.

The bit-length of d is at most $\lceil t/b \rceil \cdot s'(x,b)$ for the indices, plus $\mathrm{poly}(|x|, b, \log t)$ for the rest. The time needed by d is bounded by $\mathrm{poly}(|x|, b, \log t)$. Thus

$$\mathrm{KT}(y) \leq \lceil t/b \rceil \cdot s'(x,b) + \mathrm{poly}(|x|, b, \log t) \tag{3.7}$$

$$\leq t/b \cdot s'(x,b) + \mathrm{poly}(|x|, b, \log t) \tag{3.8}$$

where we used the fact that $s'(x,b) \leq \mathrm{poly}(|x|, b)$. The lemma follows by choosing $b = \lceil t^\alpha \rceil$. ■

3.3.2 Known Number of Automorphisms

We generalize the case of rigid graphs to graphs for which we know the size of their automorphism groups. Specifically, in addition to the two input graphs G_0 and G_1, we are also given numbers N_0, N_1 where $N_i \doteq n!/|\mathrm{Aut}(G_i)|$. Note that if $N_0 \neq N_1$, we can right away conclude that $G_0 \not\cong G_1$. Nevertheless, we do not assume that $N_0 = N_1$ as the analysis of the case $N_0 \neq N_1$ will be useful in Section 3.3.3.

The reduction is the same as in Section 3.3.1 with the correct interpretation of s. The main difference lies in the analysis, where we need to accommodate for the loss in entropy that comes from having multiple automorphisms.

Let $s_i \doteq \log(N_i)$ be the entropy in a random permutation of G_i. Set $s \doteq \min(s_0, s_1)$, and $\theta \doteq t(s + \frac{1}{2})$. In the nonisomorphic case the min-entropy of y is at least $t(s+1)$, so $\mathrm{KT}(y) > \theta$

with high probability. In the isomorphic case we upper bound $\mathrm{KT}(y)$ by about ts. Unlike the rigid case, we can no longer afford to encode an entire permutation for each permuted copy of G_0; we need a replacement for the Lehmer code. The following encoding, applied to $\Gamma = \mathrm{Aut}(G)$, suffices to complete the argument from Section 3.3.1.

Lemma 3.9. *For every subgroup Γ of S_n there exists an indexing of the cosets of Γ that is uniformly decodable in polynomial time when Γ is given by a list of generators.*

We prove Lemma 3.9 in Section 3.7 as a corollary to a more general lemma that gives, for each $\Gamma \leq H \leq S_n$, an efficiently computable indexing for the cosets of Γ in H.

Remark 3.10. Before we continue towards Theorem 3.6, we remark that the above ideas yield an alternate proof that $\mathrm{GI} \in \mathrm{BPP}^{\mathrm{MKTP}}$ (and hence that $\mathrm{GI} \in \mathrm{RP}^{\mathrm{MKTP}}$). This weaker result was already obtained in [AD17] along the well-trodden path discussed in Section 3.1; this remark shows how to obtain it using our new approach.

The key observation is that in both the isomorphic and the nonisomorphic case, with high probability $\mathrm{KT}(y)$ stays away from the threshold θ by a growing margin. Moreover, the above analysis allows us to efficiently obtain high-confidence approximations of θ to within any constant using sampling and queries to the MKTP oracle.

More specifically, for $i \in \{0,1\}$, let $\widetilde{y}_i$ denote the concatenation of $\widetilde{t}$ independent samples from R_{G_i}. Our analysis shows that $\mathrm{KT}(\widetilde{y}_i) \leq \widetilde{t}s_i + \widetilde{t}^{1-\alpha_0}n^c$ always holds, and that $\mathrm{KT}(\widetilde{y}_i) \geq \widetilde{t}s_i - \widetilde{t}^{1-\alpha_0}n^c$ holds with high probability. Thus, $\widetilde{s}_i \doteq \mathrm{KT}(\widetilde{y}_i)/\widetilde{t}$ approximates s_i with high confidence to within an additive deviation of $n^c/\widetilde{t}^{\alpha_0}$. Similarly, $\widetilde{s} \doteq \min(\widetilde{s}_0, \widetilde{s}_1)$ approximates s to within the same deviation margin, and $\widetilde{\theta} \doteq t(\widetilde{s} + \frac{1}{2})$ approximates θ to within an additive deviation of $tn^c/\widetilde{t}^{\alpha_0}$. The latter bound can be made less than 1 by setting $\widetilde{t}$ to a sufficiently large polynomial in n and t. Moreover, all these estimates can be computed in time $\mathrm{poly}(\widetilde{t}, n)$ with access to MKTP as MKTP enables us to evaluate KT efficiently. ◂

3.3.3 Probably-Correct Underestimators for the Number of Automorphisms

The reason the $\mathrm{BPP}^{\mathrm{MKTP}}$-algorithm in Remark 3.10 can have false negatives is that the approximation $\widetilde{\theta}$ to θ may be too small. Knowing the quantities $N_i \doteq n!/|\mathrm{Aut}(G_i)|$ exactly allows us to compute θ exactly and thereby obviates the possibility of false negatives. In fact, it suffices to compute overestimates for the quantities N_i which are correct with non-negligible probability. We capture this notion formally as follows:

Definition 3.11 (Probably-Correct Overestimator). Let $g : \Omega \to \mathbb{R}$ be a function, and M a randomized algorithm that, on input $\omega \in \Omega$, outputs a value $M(\omega) \in \mathbb{R}$. We say that M is a *probably-correct overestimator* for g if, for every $\omega \in \Omega$, $M(\omega) = g(\omega)$ holds with probability at least $1/\mathrm{poly}(|\omega|)$, and $M(\omega) > g(\omega)$ otherwise. A *probably-correct underestimator* for g is defined similarly by reversing the inequality. ◂

Note that, for any probably-correct overestimator (underestimator), taking the minimum (maximum) among $\mathrm{poly}(|\omega|)$ independent runs yields the correct value with probability $1 - 2^{-\mathrm{poly}(|\omega|)}$.

We are interested in the case where $g(G) = n!/|\mathrm{Aut}(G)|$. Assuming this g on a given class of graphs Ω has a probably-correct overestimator M computable in randomized polynomial time with an MKTP oracle, we argue that GI on Ω reduces to MKTP in randomized polynomial time without false negatives.

To see this, consider the algorithm that, on input a pair (G_0, G_1) of n-vertex graphs, computes $\widetilde{N}_i = M(G_i)$ as estimates of the true values $N_i = n!/|\mathrm{Aut}(G_i)|$, and then runs the algorithm from Section 3.3.2 using the estimates $\widetilde{N}_i$.

- In the case where G_0 and G_1 are not isomorphic, if both estimates $\widetilde{N}_i$ are correct, then the algorithm detects $G_0 \not\cong G_1$ with high probability.

- In the case where $G_0 \equiv G_1$, if $\widetilde{N}_i = N_i$ we showed in Section 3.3.2 that the algorithm always declares G_0 and G_1 to be isomorphic. Moreover, increasing θ can only decrease the probability of a false negative. As the computed threshold θ increases as a function of $\widetilde{N}_i$, and the estimate $\widetilde{N}_i$ is always at least as large as N_i, it follows that G_0 and G_1 are always declared isomorphic.

3.3.4 Arbitrary Graphs

A probably-correct overestimator for the function $G \mapsto n!/|\mathrm{Aut}(G)|$ on *any* graph G can be computed in randomized polynomial time with access to MKTP. The process is described in full detail in Section 3.1, based on a $\mathrm{BPP}^{\mathrm{MKTP}}$ algorithm for GI (taken from Remark 3.10 or from [AD17]). This means that the setting of Section 3.3.3 is actually the general one. The only difference is that we no longer obtain a mapping reduction from GI to MKTP, but an oracle reduction: We still make use of (3.1), but we need more queries to MKTP in order to set the threshold θ.

This shows that $\mathrm{GI} \in \mathrm{coRP}^{\mathrm{MKTP}}$. As $\mathrm{GI} \in \mathrm{RP}^{\mathrm{MKTP}}$ follows from the known search-to-decision reduction for GI, this concludes the proof of Theorem 3.6 that $\mathrm{GI} \in \mathrm{ZPP}^{\mathrm{MKTP}}$.

3.4 Estimating the Entropy of Flat Samplable Distributions

In this section we develop a key ingredient in extending Theorem 3.6 from GI to other isomorphism problems that fall within the framework presented in Section 3.1, namely efficient near-optimal encodings of cosets of automorphism groups. More generally, the encoding scheme works well for any samplable distribution that is flat or almost flat. It allows us to probably-approximately-correctly underestimate the entropy of such distributions with the help of an oracle for MKTP.

We first develop the encoding, which only requires the existence of a sampler from strings of polynomial length. The length of the encoding is roughly the max-entropy of the distribution, which is the information-theoretic optimum for flat distributions.

The lemma is stated in terms of a random variable that samples the distribution. Recall from Section 3.2 that a random variable R samples a distribution p from length ℓ when R has domain $\{0,1\}^\ell$, and p is identical to the distribution of $R(\sigma)$ with σ drawn uniformly at random from its domain.

Lemma 3.12 (Encoding Lemma). *Consider an ensemble $\{R_x\}$ of random variables that sample distributions with max-entropy $s(x)$ from length $\mathrm{poly}(|x|)$. Each R_x has an encoding of length $s(x) + \log s(x) + O(1)$ that is decodable by polynomial-size circuits.*

To see how the Encoding Lemma performs, let us apply it to the setting of GI. Consider the random variable R_G mapping a permutation $\pi \in S_n$ to $\pi(G)$. The induced distribution is flat and has entropy $s = \log(n!/|\mathrm{Aut}(G)|)$, and each $\pi \in S_n$ can be sampled from strings of length $O(n \log n)$. The Encoding Lemma thus yields an encoding of length $s + \log s + O(1)$ that is efficiently decodable. The bound on the length is worse than Lemma 3.9's bound of $\lceil s \rceil$, but will still be sufficient for the generalization of Theorem 3.6 and yield the result for GI.

We prove the Encoding Lemma using hashing. Here is the idea. Consider a random hash function $h : \{0,1\}^\ell \to \{0,1\}^m$ where ℓ denotes the length of the strings in the domain of R_x for a given x, and m is set slightly below $\ell - s$. For any fixed outcome y of R_x, there is a positive constant probability that no more than about $2^\ell/2^m \approx 2^s$ of all samples $\sigma \in \{0,1\}^\ell$ have $h(\sigma) = 0^m$, and at least one of these also satisfies $R_x(\sigma) = y$. Let us say that h *works for* y when both those conditions hold. In that case—ignoring efficiency considerations—about s bits of information are sufficient to recover a sample σ_y satisfying $R_x(\sigma_y) = y$ from h.

Now a standard probabilistic argument shows that there is a sequence $h_1, h_2, \ldots$ of $O(s)$ hash functions such that for every possible outcome y, there is at least one h_i that works for y. Given such a sequence, we can encode each outcome y as the index i of a hash function

h_i that works for y, and enough bits of information that allow us to *efficiently* recover σ_y given h_i. We show that $s + O(1)$ bits suffice for the standard linear-algebraic family of hash functions. The resulting encoding has length $s + \log(s) + O(1)$ and is decodable by circuits of polynomial size.

Proof of the Encoding Lemma. Recall that a family $\mathcal{H}_{\ell,m}$ of functions $\{0,1\}^\ell \to \{0,1\}^m$ is *universal* if for any two distinct $\sigma_0, \sigma_1 \in \{0,1\}^\ell$, the distributions of $h(\sigma_0)$ and $h(\sigma_1)$ for a uniform choice of $h \in \mathcal{H}_{\ell,m}$ are independent and uniform over $\{0,1\}^m$. We make use of the specific universal family $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ that consists of all functions of the form $\sigma \mapsto U\sigma + v$, where U is a binary $(m \times \ell)$-matrix, v is a binary column vector of dimension ℓ, and σ is also viewed as a binary column vector of dimension ℓ [CW79]. Uniformly sampling from $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ means picking U and v uniformly at random.

Claim 3.13. *Let $\ell, m \in \mathbb{N}$ and $s \in \mathbb{R}$.*

1. *For every universal family $\mathcal{H}_{\ell,m}$ with $m = \ell - \lceil s \rceil - 2$, and for every $S \subseteq \{0,1\}^\ell$ with $|S| \geq 2^{\ell - s}$,*

$$\Pr\left[(\exists \sigma \in S)\, h(\sigma) = 0^m \text{ and } |h^{-1}(0^m)| \leq 2^{\lceil s \rceil + 3}\right] \geq \frac{1}{4}, \tag{3.9}$$

   where the probability is over a uniformly random choice of $h \in \mathcal{H}_{\ell,m}$.

2. *The sets $h^{-1}(0^m)$ have indexings that are uniformly decodable in time polynomial in ℓ and m, where h ranges over $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$.*

Assume for now that the claim holds, and let us continue with the proof of the lemma.

Fix an input x, and let $\ell = \ell(x)$ and $s = s(x)$. Consider the family $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ with $m = \ell - \lceil s \rceil - 2$. For each outcome y of R_x, let S_y consist of the strings $\sigma \in \{0,1\}^\ell$ for which $R_x(\sigma) = y$. Since the distribution induced by R_x has max-entropy s, a fraction at least $1/2^s$ of the strings in the domain of R_x map to y. It follows that $|S_y| \geq 2^{\ell - s}$.

A hash function $h \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ *works* for y if there is some $\sigma \in S_y$ with $h(\sigma) = 0^m$ and $|h^{-1}(0^m)| \leq 2^{\lceil s \rceil + 3}$. By the first part of Claim 3.13, the probability that a random $h \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$

works for a fixed y is at least $1/4$. If we now pick $3\lceil s\rceil$ hash functions independently at random, the probability that none of them work for y is at most $(3/4)^{3\lceil s\rceil} < 1/2^s$. Since there are at most 2^s distinct outcomes y, a union bound shows that there exists a sequence of hash functions $h_1, h_2, \ldots, h_{3\lceil s\rceil} \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ such that for every outcome y of R_x there exists $i_y \in [3\lceil s\rceil]$ such that h_{i_y} works for y.

The encoding works as follows. Let $D^{(\mathrm{lin})}$ denote the uniform decoding algorithm from part 2 of Claim 3.13 such that $D^{(\mathrm{lin})}(h,\cdot)$ decodes the set $h^{-1}(0^m)$. For each outcome y of R_x, let $j_y \in [2^{\lceil s\rceil+3}]$ be such that $D^{(\mathrm{lin})}(h_{i_y}, j_y) = \sigma_y \in S_y$. Such a j_y exists since h_{i_y} works for y. Let $k_y = 2^{\lceil s\rceil+3} i_y + j_y$. Given $h_1, h_2, \ldots, h_{3\lceil s\rceil}$ and ℓ and m as auxiliary information, we can decode σ_y from k_y by parsing out i_y and j_y, extracting h_{i_y} from the auxiliary information, and running $D^{(\mathrm{lin})}(h_{i_y}, j_y)$. This gives an encoding for R_x of length $\lceil s\rceil + 3 + \lceil \log(3\lceil s\rceil)\rceil = s + \log s + O(1)$ that can be decoded in time $\mathrm{poly}(|x|)$ with the hash functions as auxiliary information. As each hash function can be described using $(\ell+1)m$ bits and there are $3\lceil s\rceil \le \mathrm{poly}(|x|)$ many of them, the auxiliary information consists of no more than $\mathrm{poly}(|x|)$ bits. Hard-wiring it yields a decoder circuit of size $\mathrm{poly}(|x|)$. ∎

For completeness we argue Claim 3.13.

Proof of Claim 3.13. For part 1, let $m = \ell - \lceil s\rceil - 2$, and consider the random variables $X \doteq |h^{-1}(0^m) \cap S|$ and $Y \doteq |h^{-1}(0^m)|$. Because of universality we have that $\mathbb{V}(X) \le \mathbb{E}(X) = |S|/2^m$, and by the choice of parameters $|S|/2^m \ge 4$. By Chebyshev's inequality

$$\Pr(X = 0) \le \Pr(|X - \mathbb{E}(X)| \ge \mathbb{E}(X)) \le \frac{\mathbb{V}(X)}{(\mathbb{E}(X))^2} \le \frac{1}{\mathbb{E}(X)} \le \frac{1}{4}. \tag{3.10}$$

We have that $\mathbb{E}(Y) = 2^\ell/2^m = 2^{\lceil s\rceil+2}$. By Markov's inequality

$$\Pr(Y \ge 2^{\lceil s\rceil+3}) = \Pr(Y \ge 2\,\mathbb{E}(Y)) \le \frac{1}{2}. \tag{3.11}$$

A union bound shows that

$$\Pr(X = 0 \text{ or } Y \ge 2^{\lceil s\rceil+3}) \le \frac{1}{4} + \frac{1}{2}, \tag{3.12}$$

from which part 1 follows.

For part 2, note that if $|h^{-1}(0^m)| > 0$ then $|h^{-1}(0^m)| = 2^{\ell-r}$ where r denotes the rank of U. In that case, given U and v, we can use Gaussian elimination to find binary column vectors $\hat{\sigma}$ and $\sigma_1, \sigma_2, \ldots, \sigma_{\ell-r}$ such that $U\hat{\sigma} + v = 0^m$ and the σ_i's form a basis for the kernel of U. On input $j \in [2^{\ell-r}]$, the decoder outputs $\hat{\sigma} + \sum_{i=1}^{\ell-r} j_i \sigma_i$, where $\sum_{i=1}^{\ell-r} j_i 2^{i-1}$ is the binary expansion of $j-1$. The image of the decoder is exactly $h^{-1}(0^m)$. As the decoding process runs in time $\mathrm{poly}(\ell, m)$ when given U and u, this gives the desired indexing. ■

The first part of Claim 3.13 is commonly used, e.g., for randomness extraction in cryptography. The combination of the two parts of Claim 3.13 seems to have found fewer applications. [PP10] applies them in a similar way as we do (but with a single hash function), namely to boost the success probability of randomized circuits that decide CircuitSAT as a function of the number of input variables.[3]

Remark 3.14. The proof of the Encoding Lemma shows a somewhat more general result: For any ensemble $\{R_x\}$ of random variables whose domains consist of strings of length $\mathrm{poly}(|x|)$, and for *any* bound $s(x)$, the set of outcomes of R_x with probability at least $1/2^{s(x)}$ has an encoding of length $s(x) + \log s(x) + O(1)$ that is decodable by a circuit of size $\mathrm{poly}(|x|)$. In the case of flat distributions of entropy $s(x)$ that set contains all possible outcomes. ◂

In combination with the Blocking Lemma, the Encoding Lemma yields upper bounds on KT-complexity in the case of distributions p that are samplable by polynomial-size circuits. More precisely, if y is the concatenation of t samples from p, we can essentially upper bound the amortized KT-complexity $\mathrm{KT}(y)/t$ by the max-entropy of p. On the other hand,

[3]More precisely, suppose there exists a randomized circuit family A of size $f(n, m)$ that decides CircuitSAT without false positives on instances consisting of circuits C with n input variables and of description length m such that the probability of success is at least $1/2^{\alpha n}$. Applying our encoding to the set of random bit sequences that make A accept on a positive instance C, and hard-wiring the input C into the circuit A, yields an equivalent instance C' on αn variables of size $f(n, m) + \mu(D)$, where $\mu(D)$ denotes the circuit size of D. Applying A to the description of this new circuit C' yields a randomized circuit A' to decide whether C is satisfiable without false positives. For the linear-algebraic family of hash functions, A' has size $O(f(n, m)\mathrm{polylog}(f(n, m)))$. Its success probability is at least $1/2^{\alpha^2 n}$, which is larger than $1/2^{\alpha n}$ when $\alpha < 1$.

Proposition 3.4 shows that if the samples are picked independently at random, with high probability $\mathrm{KT}(y)/t$ is not much less than the min-entropy of p. Thus, in the case of flat distributions, $\mathrm{KT}(y)/t$ is a good *probably-approximately-correct underestimator* for the entropy, a notion formally defined as follows.

Definition 3.15 (Probably-Approximately-Correct Underestimator). Let $g : \Omega \to \mathbb{R}$ be a function, and M a randomized algorithm that, on input $\omega \in \Omega$, outputs a value $M(\omega) \in \mathbb{R}$. We say that M is a *probably-approximately-correct underestimator* (or *pac underestimator*) for g with deviation Δ if, for every $\omega \in \Omega$, $|M(\omega) - g(\omega)| \leq \Delta$ holds with probability at least $1/\mathrm{poly}(|\omega|)$, and $M(\omega) < g(\omega)$ otherwise. A *probably-approximately-correct overestimator* (or *pac overestimator*) for g is defined similarly, by reversing the last inequality. ◂

Similar to the case of probably-correct under-/overestimators, we can boost the confidence level of a pac under-/overestimator from $1/\mathrm{poly}(|\omega|)$ to $1 - 2^{-\mathrm{poly}(|\omega|)}$ by taking the max/min of $\mathrm{poly}(|\omega|)$ independent runs.

More generally, we argue that the amortized KT-complexity of samples yields a good pac underestimator for the entropy when the distribution is *almost* flat, i.e., the difference between the max- and min-entropy is small. As KT can be evaluated efficiently with oracle access to MKTP, pac underestimating the entropy of such distributions reduces to MKTP.

Corollary 3.16 (Entropy Estimator Corollary). *Let $\{p_x\}$ be an ensemble of distributions such that p_x is supported on strings of the same length $\mathrm{poly}(|x|)$. Consider a randomized process that on input x computes $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from p_x. If p_x is samplable by circuits of polynomial size, then for t a sufficiently large polynomial in $|x|$, $\mathrm{KT}(y)/t$ is a pac underestimator for the entropy of p_x with deviation $\Delta(x) + o(1)$, where $\Delta(x)$ is the difference between the min- and max-entropies of p_x.*

Proof. Since the entropy lies between the min- and max-entropies, it suffices to show that $\mathrm{KT}(y)/t$ is at least the min-entropy of p_x with high probability, and is always at most the

max-entropy of p_x (both up to $o(1)$ terms) when t is a sufficiently large polynomial. The lower bound follows from Proposition 3.4. It remains to establish the upper bound.

Let $\{R_{x,\delta}\}$ be the ensemble of random variables witnessing the samplability of $\{p_x\}$ by circuits of polynomial size, and let $s(x)$ denote the max-entropy of p_x. The Blocking Lemma allows us to bound $\mathrm{KT}(y)$ by giving an encoding for random variables whose support contains the b-tuples of samples from p_x. Let $R'_{x,b}$ denote the b-fold Cartesian product of $R_{x,1/b}$. $R'_{x,b}$ induces a distribution that approximates to within a factor of $(1+1/b)^b = O(1)$ the distribution of the b-fold Cartesian product of p_x, which is a distribution of max-entropy $bs(x)$. It follows that the distribution induced by $R'_{x,b}$ has min-entropy at most $bs(x)+O(1)$. Its support is exactly the b-tuples of samples from p_x. Moreover, the ensemble $\{R'_{x,b}\}$ is computable by circuits of size $\mathrm{poly}(n,b)$. By the Encoding Lemma there exists an encoding of $R'_{x,b}$ of length $bs(x)+\log b+\log s(x)+O(1)$ that is decodable by circuits of polynomial-size. The Blocking Lemma then says that there exist constants c_1 and c_2 so that for all $\alpha > 0$ and all sufficiently large n

$$\mathrm{KT}(y) \le t^{1-\alpha} \cdot (\lceil t^\alpha \rceil \cdot s(x) + \log s(x) + \alpha \log t + O(1)) + t^{\alpha c_1} \cdot n^{c_2} \tag{3.13}$$

$$\le ts(x) + t^{1-\alpha} \cdot (n^{c_0} + c_0 \log n + \alpha \log t + O(1)) + t^{\alpha c_1} \cdot n^{c_2}, \tag{3.14}$$

where we use the fact that there exists a constant c_0 such that $s(x) \le n^{c_0}$. A similar calculation as the one following equation (3.6) shows that $\mathrm{KT}(y) \le ts(x) + t^{1-\alpha_0} n^{c_0+c_2}$ for $t \ge n^c$ and n sufficiently large, where $\alpha_0 = 1/(1+c_1)$ and $c = 1 + (1+c_1)(c_0+c_2)$. Dividing both sides by t yields the claimed upper bound. ∎

3.5 Generic Isomorphism Problem

In Section 3.1 we presented a common framework for isomorphism problems and listed some instantiations in Table 3.1. In this section we state and prove a generalization of Theorem 3.6 that applies to many problems in this framework, including the ones from Table 3.1.

3.5.1 Generalization

The generalized reduction makes use of a complete invariant for the abstract universe Ω. For future reference, we define the notion with respect to a representation for an arbitrary ensemble of sets.

Definition 3.17 (Representation, Complete Invariant). Let $\{\Omega_x\}$ denote an ensemble of sets. A *representation* of the ensemble is a surjective mapping $\omega : \{0,1\}^* \to \cup_x \Omega_x$. A *complete invariant* for ω is a mapping $\nu : \{0,1\}^* \to \{0,1\}^*$ such that for all strings x, z_0, z_1 with $\omega(z_0), \omega(z_1) \in \Omega_x$

$$\omega(z_0) = \omega(z_1) \Leftrightarrow \nu(z_0) = \nu(z_1). \tag{3.15}$$

◂

$\omega(z)$ denotes the set element represented by the string z. The surjective property of a representation guarantees that every set element has at least one string representing it.

Note that for the function ν to represent a *normal form* (rather than just a complete invariant), it would need to be the case that $\omega(\nu(z)) = \omega(z)$. Although this additional property holds for all the instantiations we consider, it is not a requirement. In our setting, all that matters is that $\nu(z)$ only depends on the element $\omega(z)$ that z represents, and is different for different elements. For complexity-theoretic investigations into the difference between complete invariants and normal forms, see, e.g., [BG84a; BG84b; FG11; FH16].

We are now ready to state the generalization of Theorem 3.6.

Theorem 3.18. *Let* Iso *denote an isomorphism problem as in Definition 3.1. Consider the following conditions:*

1. [Action Sampler] *The uniform distribution on H_x is uniformly samplable in polynomial time, and the mapping $(\omega, h) \mapsto h(\omega)$ underlying the action of H_x on Ω_x is computable in* ZPP.

2. [Complete Universe Invariant] *There exists a complete invariant ν for the representation ω that is computable in* ZPP.

3. [Entropy Estimator] *There exists a probably-approximately-correct overestimator for* $(x, \omega) \mapsto \log(|H_x|/|\mathrm{Aut}(\omega)|)$ *with deviation* $1/4$ *that is computable in randomized time* $\mathrm{poly}(|x|)$ *with access to an oracle for* MKTP.

With these definitions:

(a) If conditions 1 and 2 hold, then $\mathrm{Iso} \in \mathrm{RP}^{\mathrm{MKTP}}$.

(b) If conditions 1, 2, and 3 hold, then $\mathrm{Iso} \in \mathrm{coRP}^{\mathrm{MKTP}}$.

In the case of GI, Ω denotes the universe of graphs on n vertices (represented as adjacency matrices viewed as strings of length n^2), and H the group of permutations on $[n]$ (represented as function tables). All conditions in the statement of Theorem 3.18 are met. The identity mapping can be used as the complete invariant ν in condition 2, and the probably-correct overestimator for $n!/|\mathrm{Aut}(G)|$ that we argued in Sections 3.1 and 3.3 immediately yields the pac overestimator for $\log(n!/|\mathrm{Aut}(G)|)$ required in condition 3.

Note that $\log(n!/|\mathrm{Aut}(G)|)$ equals the entropy of the distribution induced by the random variable R_G. In general, the quantity $\log(|H_x|/|\mathrm{Aut}(\omega)|)$ in condition 3 represents the entropy of $\nu(h(\omega))$ when $h \in H_x$ is picked uniformly at random.

Proof of Theorem 3.18. Let x denote an instance of length $n \doteq |x|$, defining a universe Ω, a group H that acts on Ω, and two elements $\omega_i = \omega_i(x)$ for $i \in \{0,1\}$. Both parts (a) and (b) make use of the random variables R_i for $i \in \{0,1\}$ where $R_i : H \to \{0,1\}^*$ maps $h \in H$ to $\nu(h(\omega_i))$.

Part (a) We follow the approach from [AD17]. Their argument uses Lemma 3.3, which states the existence of a randomized polynomial-time machine M with access to an MKTP oracle that, given a random sample y from the distribution induced by a circuit C, recovers

with non-negligible probability of success an input σ so that $C(\sigma) = y$. If we can model the R_i as circuits of size $\text{poly}(n)$ that take in an element h from H and output $R_i(h)$, this means that, with non-negligible probability over a random $h_0 \in H$, $M(R_0, R_0(h_0))$ outputs some h_1 so that $h_1(\omega_0) = h_0(\omega_0)$. The key observation is that when $\omega_0 \equiv \omega_1$, R_0 and R_1 induce the same distribution, and therefore, for a random element h_0, $M(R_1, R_0(h_0))$ outputs some h_1 so that $h_1(\omega_0) = h_0(\omega_0)$ with non-negligible probability probability of success. Thus Iso can be decided by trying the above a polynomial number of times, declaring $\omega_0 \equiv \omega_1$ if a trial succeeds, and declaring $\omega_0 \not\equiv \omega_1$ otherwise.

We do not know how to model the R_i exactly as circuits of size $\text{poly}(n)$, but we can do so approximately. Condition 1 implies that we can construct circuits $C_{i,\delta}$ in time $\text{poly}(n/\delta)$ that sample $h(\omega_i)$ within a factor $1+\delta$. Combined with the ZPP-computability of ν in condition 2 this means that we can construct a circuit C_ν in time $\text{poly}(n)$ such that the composed circuit $C_\nu \circ C_{i,\delta}$ samples R_i within a factor $1 + \delta$ from strings σ of length $\text{poly}(n/\delta)$. We use the composed circuits in lieu of R_i in the arguments for M above. More precisely, we pick an input σ_0 for $C_{0,\delta}$ uniformly at random, and compute $\sigma_1 = M(C_{1,\delta}, C_{0,\delta}(\sigma_0))$. Success means that $h_1(\omega_0) = h_0(\omega_0)$, where $h_i = C_{i,\delta}(\sigma_i)$. The probability of success for an approximation factor of $1 + \delta$ is at least $1/(1 + \delta)^2$ times the probability of success in the exact setting, which is $1/\text{poly}(n/\delta)$ in the isomorphic case. Fixing δ to any positive constant, a single trial runs in time $\text{poly}(n)$, success can be determined in ZPP (by the second part of condition 1), and the probability of success is at least $1/\text{poly}(n)$ in the isomorphic case. Completing the argument as in the exact setting above, we conclude that $\text{Iso} \in \text{RP}^{\text{MKTP}}$.

Part (b) We extend the argument from Section 3.3. Let $s_i \doteq \log(|H|/|\text{Aut}(\omega_i)|)$ for $i \in \{0, 1\}$, and let M be the pac overestimator from condition 3. We assume that M has been amplified such that it outputs a good estimate with probability exponentially close to 1. Condition 1 and the ZPP-computability of ν imply that the distribution induced by R_i is uniformly samplable in polynomial time, i.e., for each $i \in \{0, 1\}$ and $\delta > 0$, there is a

random variable $R_{i,\delta}$ that samples R_i within a factor $1+\delta$ from length $\mathrm{poly}(|x|/\delta)$, and that is computable in time $\mathrm{poly}(|x|/\delta)$.

Let $t \in \mathbb{N}$ and δ be parameters to be determined. On input x, the algorithm begins by computing the estimates $\widetilde{s}_i = M(x, \omega_i)$ for $i \in \{0,1\}$, and sets $\widetilde{s} \doteq \min(\widetilde{s}_0, \widetilde{s}_1)$ and $\widetilde{\theta} \doteq t(\widetilde{s} + \frac{1}{2})$. The algorithm then samples $r \in \{0,1\}^t$ uniformly, and constructs $y = (R_{r_i,\delta}(\sigma_i))_{i=1}^t$, where each σ_i is drawn independently and uniformly from $\{0,1\}^{\mathrm{poly}(n,1/\delta)}$. If $\mathrm{KT}(y) > \widetilde{\theta}$, the algorithm declares $\omega_0 \not\equiv \omega_1$; otherwise, the algorithm declares $\omega_0 \equiv \omega_1$.

Nonisomorphic Case If $\omega_0 \not\equiv \omega_1$, we need to show $\mathrm{KT}(y) > \widetilde{\theta}$ with high probability. Since $R_{i,\delta}$ samples R_i within a factor of $1+\delta$, and R_i is flat with entropy s_i, it follows that $R_{i,\delta}$ has min-entropy at least $s_i - \log(1+\delta)$, and that y is sampled from a distribution with min-entropy at least

$$t(1 + \min(s_0, s_1) - \log(1+\delta)). \tag{3.16}$$

Since M is a pac overestimator with deviation $1/4$, $|\widetilde{s}_0 - s_0| \le 1/4$ and $|\widetilde{s}_1 - s_1| \le 1/4$ with high probability. When this happens, $\widetilde{s} \le \min(s_0, s_1) + 1/4$,

$$\widetilde{\theta} \le t(\min(s_0, s_1) + 3/4), \tag{3.17}$$

and Proposition 3.4 guarantees that $\mathrm{KT}(y) > \widetilde{\theta}$ except with probability exponentially small in t as long as δ is a constant such that $1 - \log(1+\delta) > 3/4$. Such a positive constant δ exists.

Isomorphic Case If $\omega_0 \equiv \omega_1$, we need to show that $\mathrm{KT}(y) \le \widetilde{\theta}$ always holds for t a sufficiently large polynomial in n, and n sufficiently large. Recall that, since $\omega_0 \equiv \omega_1$, R_0 and R_1 induce the same distribution, so we can view y as the concatenation of t samples from R_0. Each R_0 is flat, hence has min-entropy equal to its max-entropy, and the ensemble of all R_0 (across all inputs x) is samplable by (uniform) polynomial-size circuits. The Entropy Estimator Corollary with $\Delta(x) \equiv 0$ then implies that $\mathrm{KT}(y) \le t(s_0 + o(1))$ holds whenever t is a sufficiently large polynomial in n, and n is sufficiently large. In that case, $\mathrm{KT}(y) \le$

$t(\widetilde{s}+\frac{1}{4}+o(1)) < \widetilde{\theta}$ holds because $s_0 \le \widetilde{s}+1/4$ follows from M being a pac overestimator for s_0 with deviation $1/4$. ■

Remark 3.19. The notion of efficiency in conditions 1, and 2 can be relaxed to mean the underlying algorithm is implementable by a family of polynomial-size circuits that is constructible in $\mathrm{ZPP}^{\mathrm{MKTP}}$. It is important for the argument that the circuits themselves do not have oracle access to MKTP, but it is all right for them to be constructible in $\mathrm{ZPP}^{\mathrm{MKTP}}$ rather than P or ZPP. For example, a sampling procedure that requires knowing the factorization of some number (dependent on the input x) is fine because the factorization can be computed in $\mathrm{ZPP}^{\mathrm{MKTP}}$ [ABK+06] and then can be hard-wired into the circuit.

In particular, this observation yields an alternate way to show that integer factorization being in $\mathrm{ZPP}^{\mathrm{MKTP}}$ implies that the discrete log over prime fields is in $\mathrm{ZPP}^{\mathrm{MKTP}}$ [Rud17]. Recall that an instance of the discrete log problem consists of a triple $x = (g, z, p)$, where g and z are integers, and p is a prime, and the goal is to find an integer y such that $g^y \equiv z \bmod p$, or report that no such integer exists. The search version is known to reduce to the decision version in randomized polynomial time, and the above observation shows that the decision version is in $\mathrm{ZPP}^{\mathrm{MKTP}}$. This is because computing the size of the subgroup of $\mathbb{F}_p^\times$ generated by g or z reduces to integer factorization, and can thus be computed in $\mathrm{ZPP}^{\mathrm{MKTP}}$. ◂

3.5.2 Construction of Probably-Correct Overestimators

We now discuss some generic methods to satisfy condition 3 in Theorem 3.18, i.e., how to construct a probably-approximately-correct overestimator for the quantity $\log(|H|/|\mathrm{Aut}(\omega)|)$ that is computable in $\mathrm{ZPP}^{\mathrm{MKTP}}$.

Here is the generalization of the approach we used in Section 3.3.4 in the context of GI:

1. Find a list L of elements of H that generates a subgroup $\langle L\rangle$ of $\mathrm{Aut}(\omega)$ such that $\langle L\rangle = \mathrm{Aut}(\omega)$ with high probability.

2. Pac underestimate $\log|\langle L\rangle|$ with deviation 1/8. This yields a pac underestimator for $\log|\mathrm{Aut}(\omega)|$.

3. Pac overestimate $\log|H|$ with deviation 1/8.

4. Return the result of step 3 minus the result of step 2. This gives a pac overestimator for $\log(|H|/|\mathrm{Aut}(\omega)|)$ with deviation 1/4.

Although in the setting of GI we used the oracle for MKTP only in step 1, we could use it to facilitate steps 2 and 3 as well.

The first step for GI follows from the known search-to-decision reduction. It relies on the fact that *Colored Graph Isomorphism* reduces to GI, where Colored Graph Isomorphism allows one to assign colors to vertices with the understanding that the isomorphism needs to preserve the colors. For all of the isomorphism problems in Table 3.1, finding a set of generators for the automorphism group reduces to a natural colored version of the isomorphism problem, but it is not clear whether the colored version always reduces to the regular version. The latter reduction is known for Linear Code Equivalence, but remains open for problems like Permutation Group Conjugacy and Matrix Subspace Conjugacy.

However, there is a different, *generic* way to achieve step 1 above, namely based on Lemma 3.3, i.e., the power of MKTP to efficiently invert on average any efficiently computable function.

Lemma 3.20. *Let* Iso *denote an isomorphism problem as in Definition 3.1 that satisfies conditions 1 and 2 of Theorem 3.18, and such that products and inverses in H_x are computable in* $\mathrm{BPP}^{\mathrm{MKTP}}$. *There exists a randomized polynomial-time algorithm using oracle access to* MKTP *with the following behavior: On input any instance x, and any $\omega \in \Omega_x$, the algorithm outputs a list of generators for a subgroup Γ of $\mathrm{Aut}(\omega)$ such that $\Gamma = \mathrm{Aut}(\omega)$ with probability $1 - 2^{-|x|}$.*

Proof. Consider an instance x of length $n \doteq |x|$, and $\omega \in \Omega_x$. We first argue that the uniform distribution on $\mathrm{Aut}(\omega)$ is uniformly samplable in polynomial time with oracle access to MKTP.

Let R_ω denote the random variable that maps $h \in H$ to $\nu(h(\omega))$, and let M be the machine from Lemma 3.3. As in the proof of Part 1 of Theorem 3.18, we can sample h from H uniformly (to within a small constant factor) and run $M(R_\omega, \nu(h(\omega)))$ to obtain, with non-negligible probability, some $h' \in H$ such that $h'(\omega) = h(\omega)$. In that case, $h^{-1}h'$ is an automorphism of ω, and we say the process succeeds. The key observation is the following: Since $h' = M(R_\omega, \nu(h(\omega)))$, the distribution of h' conditioned on h only depends on the coset of $\mathrm{Aut}(G)$ that h belongs to. It follows that if h were sampled perfectly uniformly then, conditioned on success, the distribution of $h^{-1}h'$ is *uniform* over $\mathrm{Aut}(\omega)$. In truth, h is sampled uniformly to within a factor $1+\delta$; in that case $h^{-1}h'$ is (conditioned on success) likewise uniform on $\mathrm{Aut}(\omega)$ to within a factor $1+\delta$ and, as argued in the proof of Theorem 3.18, the probability of success is $1/\mathrm{poly}(n/\delta)$.

We run the process many times and retain the automorphism $h^{-1}h'$ from the first successful run (if any); $\mathrm{poly}(n/\delta)$ runs suffice to obtain, with probability $1-2^{-2n}$, an automorphism that is within a factor $1+\delta$ from uniform over $\mathrm{Aut}(\omega)$. By the computability parts of conditions 1 and 2, and by the condition that products and inverses in H can be computed in $\mathrm{BPP}^{\mathrm{MKTP}}$, each trial runs in time $\mathrm{poly}(n/\delta)$. Success can be determined in ZPP as the group action is computable in ZPP. It follows that the uniform distribution on $\mathrm{Aut}(\omega)$ is uniformly samplable in polynomial time with oracle access to MKTP.

Finally, we argue that a small number of independent samples $h_1, h_2, \ldots, h_k$ for some constant $\delta > 0$ suffice to ensure that they generate all of $\mathrm{Aut}(\omega)$ with very high probability. Denote by Γ_i the subgroup of H_x generated by $h_1, \ldots, h_i$. Note that Γ_i always is a subgroup of $\mathrm{Aut}(\omega)$. For $i < k$, if Γ_i is not all of $\mathrm{Aut}(\omega)$, then $|\Gamma_i| \le |\mathrm{Aut}(\omega)|/2$. Thus, with probability at least $\frac{1}{2} \cdot \frac{1}{1+\delta}$, $h_{i+1} \notin \Gamma_i$, in which case $|\Gamma_{i+1}| \ge 2|\Gamma_i|$. For any constant $\delta > 0$, it follows that $k \ge \Theta(n + \log|\mathrm{Aut}(\omega)|) = O(\mathrm{poly}(n))$ suffices to guarantee that $\Gamma_k = \mathrm{Aut}(\omega)$ with probability

at least $1 - 2^{-2n}$. The lemma follows. ∎

The second step for GI followed from the ability to efficiently compute the order of permutation groups exactly. Efficient exact algorithms (perhaps with access to an oracle for MKTP) are known for larger classes of groups in specific representations (see, e.g., [BBS09] and subsequent work for matrix groups), but not for all. We work around this by showing how to *generically* pac underestimate $\log|\langle L\rangle|$ with small deviation (step 2), namely under the prior conditions that only involve H, and the additional condition of a ZPP-computable complete invariant ζ for H.

The construction hinges on the Entropy Estimator Corollary and viewing $\log|\langle L\rangle|$ as the entropy of the uniform distribution p_L on $\langle L\rangle$.

(α) Provided that p_L is samplable by circuits of polynomial size, the corollary allows us to pac underestimate $\log|\langle L\rangle|$ as $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from p_L.

(β) If we are able to uniformly sample $\{p_L\}$ *exactly* in polynomial time (possibly with access to an oracle for MKTP), then we can evaluate the estimator $\mathrm{KT}(y)/t$ in polynomial time with access to MKTP. This is because the oracle for MKTP lets us evaluate KT in polynomial time.

Thus, if we were able to uniformly sample $\{p_L\}$ *exactly* in polynomial time, we'd be done. We do not know how to do that, but we can do it *approximately*, which we argue is sufficient.

The need for a ZPP-computable complete invariant comes in when representing the abstract group elements as strings. In order to formally state the requirement, we make the underlying representation of group elements explicit; we denote it by η.

Lemma 3.21. *Let $\{H_x\}$ be an ensemble of groups. Suppose that the ensemble has a representation η such that the uniform distribution on H_x is uniformly samplable in polynomial-time, products and inverses in H_x are computable in* ZPP*, and there exists a* ZPP*-computable complete invariant for η. Then for any list L of elements of H_x, the logarithm of the order of the*

group generated by L, i.e., $\log|\langle L\rangle|$*, can be pac underestimated with any constant deviation* $\Delta > 0$ *in randomized time* $\mathrm{poly}(|x|,|L|)$ *with oracle access to* MKTP.

Proof. Let ζ be the ZPP-computable complete invariant for η. For each list L of elements of H_x, let p_L denote the distribution of $\zeta(h)$ when h is picked uniformly at random from $\langle L\rangle$. Note that p_L is flat with entropy $s = \log|\langle L\rangle|$.

Claim 3.22. *The ensemble of distributions* $\{p_L\}$ *is uniformly samplable in polynomial time.*

For every constant $\delta > 0$, the claim yields a family of random variables $\{R_{L,\delta}\}$ computable uniformly in polynomial time such that $R_{L,\delta}$ induces a distribution $p_{L,\delta}$ that approximates p_L to within a factor $1+\delta$. Note that the min-entropy of $p_{L,\delta}$ is at least $s - \log(1+\delta)$, and the max-entropy of $p_{L,\delta}$ at most $s + \log(1+\delta)$, thus their difference is no more than $2\log(1+\delta)$.

Let $M_\delta(L)$ denote $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from $p_{L,\delta}$.

(α) The Entropy Estimator Corollary guarantees that for any sufficiently large polynomial t, M_δ is a pac underestimator for the entropy of $p_{L,\delta}$ with deviation $2\log(1+\delta)+o(1)$, and thus a pac underestimator for $s = \log|\langle L\rangle|$ with deviation $3\log(1+\delta)+o(1)$.

(β) For any polynomial t, we can compute M_δ in polynomial time with access to an oracle for MKTP. This is because $R_{L,\delta}$ enables us to generate y in polynomial time. We then use the oracle for MKTP to compute $\mathrm{KT}(y)$ exactly, and divide by t.

Thus, M_δ meets all the requirements for our estimator as long as $3\log(1+\delta) < \Delta$, which holds for some positive constant δ.

This completes the proof of Lemma 3.21 modulo the proof of the claim. ∎

The proof of Claim 3.22 relies on the notion of *Erdős–Rényi generators*. A list of generators $L = (h_1, \ldots, h_k)$ is said to be Erdős–Rényi with factor $1+\delta$ if a random subproduct of L approximates the uniform distribution on $\langle L\rangle$ within a factor $1+\delta$, where a random

subproduct is obtained by picking $r_i \in \{0,1\}$ for each $i \in [k]$ uniformly at random, and outputting $h_1^{r_1} h_2^{r_2} \cdots h_k^{r_k}$.

Proof of Claim 3.22. By definition, if L happens to be Erdős–Rényi with factor $1+\delta$, then p_L can be sampled to within a factor $1+\delta$ with fewer than $|L|$ products in H_x.

Erdős and Rényi [ER65] showed that, for any finite group Γ, with high probability, a list of $\mathrm{poly}(\log|\Gamma|, \log(1/\delta))$ random elements of Γ form an Erdős–Rényi list of generators with factor $1+\delta$. For $\Gamma = \langle L \rangle$, this gives a list L' for which we can sample $p_{L'} = p_L$. By hard-wiring the list L' into the sampler for $p_{L'}$, it follows that p_L is samplable by circuits of size $\mathrm{poly}(\log|\langle L\rangle|, \log(1/\delta)) \le \mathrm{poly}(|L|/\delta)$.

As for *uniformly* sampling $\{p_L\}$ in polynomial time, Theorem 1.1 in [Bab91] gives a randomized algorithm that, given L, generates a list L' of elements from $\langle L \rangle$ that, with probability $1-\varepsilon$, are Erdős–Rényi with factor $1+\delta$. The running time of the algorithm is $\mathrm{poly}(|x|, |L|, \log(1/\delta), \log(1/\varepsilon))$ assuming products and inverses in H_x can be computed in ZPP. For $\varepsilon = \delta/|\langle L\rangle|$, the overall distribution of a random subproduct of L' is within a factor $1+2\delta$ from p_L, and can be generated in time $\mathrm{poly}(|x|, |L|, \log(1/\delta)) \le \mathrm{poly}(|x|, |L|, 1/\delta)$. As δ can be an arbitrary positive constant, it follows that p_L is uniformly samplable in polynomial time. ∎

Following the four steps listed at the beginning of this section, we can replace condition 3 in Theorem 3.18 by the conditions of Lemma 3.20 (for step 1), those of Lemma 3.21 (for step 2), and the existence of an estimator for the size $|H|$ of the sample space as stated in step 3. This gives the following result:

Theorem 3.23. *Let* Iso *denote an isomorphism problem as in Definition 3.1. Suppose that the ensemble $\{H_x\}$ has a representation η such that conditions 1 and 2 of Theorem 3.18 hold as well as the following additional conditions:*

4. [Group Operations] *Products and inverses in H_x are computable in* ZPP.

5. [Sample Space Estimator] *The map* $x \mapsto |H_x|$ *has a pac overestimator with deviation* $1/8$ *computable in* $\mathrm{ZPP}^{\mathrm{MKTP}}$.

6. [Complete Group Invariant] *There exists a complete invariant* ζ *for the representation* η *that is computable in* ZPP.

Then $\mathrm{Iso} \in \mathrm{ZPP}^{\mathrm{MKTP}}$.

As was the case for Theorem 3.18, the conditions of Theorem 3.23 can be satisfied in a straightforward way for GI. The representation η of the symmetric groups S_n meets all the requirements that only involve the underlying group: uniform samplability as in the first part of condition 1, efficient group operations as in condition 4, the sample space size $|H| = |S_n| = n!$ can be computed efficiently (condition 5), and the identity mapping can be used as the complete group invariant ζ (condition 6). The efficiency of the action (the second part of condition 1) and condition 2 about a complete universe invariant are also met in the same way as before.

We remark that Claim 3.22 can be used to show that the uniform distribution on H_x is uniformly samplable in polynomial time (the first part of condition 1), provided a set of generators for H_x can be computed in ZPP. This constitutes another use of [Bab91, Theorem 1.1].

On the other hand, the use of [Bab91, Theorem 1.1] in the proof of Theorem 3.23 can be eliminated. Referring to parts (α) and (β) in the intuition and proof of Lemma 3.21, we note the following:

(α) The first part of the proof of Claim 3.22 relies on [ER65] but not on [Bab91, Theorem 1.1]. It shows that p_L is samplable by polynomial-size circuits, which is sufficient for the Entropy Estimator Corollary to apply and show that $M_\delta(L) = \mathrm{KT}(y)/t$ is a pac underestimator for $\log|\langle L\rangle|$ with deviation $3\log(1+\delta) + o(1)$, where y is the concatenation of t independent samples from $p_{L,\delta}$ for a sufficiently large polynomial t.

(β) Specialized to the case where $\langle L \rangle = \mathrm{Aut}(\omega)$, the first part of the proof of Lemma 3.20 shows that, for any constant $\delta > 0$, $p_{L,\delta}$ is *uniformly* samplable in polynomial time with access to an oracle for MKTP. Once we have generated y with the help of MKTP, we use MKTP once more to evaluate $\mathrm{KT}(y)$ and output $M_\delta(L) = \mathrm{KT}(y)/t$.

This way, for any constant $\delta > 0$ we obtain a pac underestimator M_δ for $\log|\mathrm{Aut}(\omega)|$ with deviation $3\log(1+\delta) + o(1)$ that is computable in polynomial time with access to MKTP.

This alternate construction replaces steps 1 and 2 in the outline from the beginning of this section. The resulting alternate proof of Theorem 3.23 is more elementary (as it does not rely on [Bab91, Theorem 1.1]) but does not entirely follow the approach we used for GI of first finding a list L of elements that likely generates $\mathrm{Aut}(\omega)$ (and never generates more) and then determining the size of the subgroup generated by L.

Remark 3.24. Remark 3.19 on relaxing the efficiency requirement in conditions 1 and 2 of Theorem 3.18 extends similarly to Theorem 3.23. For Theorem 3.23, it suffices that all the computations mentioned in conditions 1, 2, 4, and 6 be do-able by $\mathrm{ZPP}^{\mathrm{MKTP}}$-constructible ordinary circuits. ◂

3.6 Instantiations

In this section we argue that Theorem 3.23 applies to the example isomorphism problems listed in Table 3.1 (other than GI, which we covered in Section 3.3). We describe each problem, provide some background, and show that the conditions of Theorem 3.23 hold, thus proving that the problem is in $\mathrm{ZPP}^{\mathrm{MKTP}}$.

Linear Code Equivalence A *linear code* over the finite field $\mathbb{F}_q$ is a d-dimensional linear subspace of $\mathbb{F}_q^n$ for some n. Two such codes are (permutationally) *equivalent* if there is a permutation of the n coordinates that makes them equal as subspaces.

Linear Code Equivalence is the problem of deciding whether two linear codes are equivalent, where the codes are specified as the row-span of a $d \times n$ matrix (of rank d), called a *generator matrix*. Note that two different inputs may represent the same code. There exists a mapping reduction from GI to Linear Code Equivalence over any field [PR97; Gro12]; Linear Code Equivalence is generally thought to be harder than GI.

In order to cast Code Equivalence in our framework, we consider the family of actions $(S_n, \Omega_{n,d,q})$ where $\Omega_{n,d,q}$ denotes the linear codes of length n and dimension d over $\mathbb{F}_q$, and S_n acts by permuting the coordinates. To apply Theorem 3.23, as the underlying group is S_n, we only need to check the efficiency of the action (second part of condition 1) and the complete universe invariant (condition 2). The former holds because the action only involves swapping columns in the generator matrix. For condition 2 we can define $\nu(z)$ to be the reduced row echelon form of z. This choice works because two generator matrices define the same code iff they have the same reduced row echelon form, and it can be computed in polynomial time.

Corollary 3.25. *Linear Code Equivalence is in* $\mathrm{ZPP}^{\mathrm{MKTP}}$.

Permutation Group Conjugacy Two permutation groups $\Gamma_0, \Gamma_1 \leq S_n$ are *conjugate* (or permutationally isomorphic) if there exists a permutation $\pi \in S_n$ such that $\Gamma_1 = \pi\Gamma_0\pi^{-1}$; such a π is called a conjugacy.

The *Permutation Group Conjugacy* problem is to decide whether two subgroups of S_n are conjugate, where the subgroups are specified by a list of generators. The problem is known to be in $\mathrm{NP} \cap \mathrm{coAM}$, and is at least as hard as Linear Code Equivalence. Currently the best known algorithm runs in time $2^{O(n)}$ [Wie20].

Casting Permutation Group Conjugacy in the framework is similar to before: S_n acts on the subgroup by conjugacy. The action is computable in polynomial time (second part of condition 1) as it only involves inverting and composing permutations. It remains to check condition 2. Note that there are many different lists that generate the same subgroup. We

make use of the normal form provided by the following lemma.

Lemma 3.26. *There is a* poly(n)*-time algorithm* ν *that takes as input a list* L *of elements of* S_n*, and outputs a list of generators for the subgroup generated by the elements in* L *such that for any two lists* L_0, L_1 *of elements of* S_n *that generate the same subgroup,* $\nu(L_0) = \nu(L_1)$*.*

The normal form from Lemma 3.26 was known to some experts (Babai, personal communication); for completeness we provide a proof in Section 3.7. By Theorem 3.23 we conclude:

Corollary 3.27. *Permutation Group Conjugacy is in* $\mathrm{ZPP}^{\mathrm{MKTP}}$*.*

Matrix Subspace Conjugacy A *linear matrix space* over $\mathbb{F}_q$ is a d-dimensional linear subspace of $n \times n$ matrices. Two such spaces V_0 and V_1 are *conjugate* if there is an invertible $n \times n$ matrix X such that $V_1 = XV_0X^{-1} \doteq \{X \cdot M \cdot X^{-1} : M \in V_0\}$, where "$\cdot$" represents matrix multiplication.

Matrix Subspace Conjugacy is the problem of deciding whether two linear matrix spaces are conjugate, where the spaces are specified as the linear span of d linearly independent $n \times n$ matrices. Many other isomorphism problems reduce to Matrix Subspace Conjugacy, and in particular there exist mapping reductions from GI and Linear Code Equivalence to Matrix Subspace Conjugacy [Gro12; GQ21]. Matrix Subspace Conjugacy is generally thought to be harder than Linear Code Equivalence.

In order to cast Matrix Subspace Conjugacy in our framework, we consider the family of actions $(\mathrm{GL}_n(\mathbb{F}_q), \Omega_{n,d,q})$ where $\mathrm{GL}_n(\mathbb{F}_q)$ denotes the n-by-n general linear group over $\mathbb{F}_q$ (consisting of all invertible n-by-n matrices over $\mathbb{F}_q$ with multiplication as the group operation), $\Omega_{n,d,q}$ represents the set of d-dimensional subspaces of $\mathbb{F}_q^{n \times n}$, and the action is by conjugation. As was the case with Linear Code Equivalence, two inputs may represent the same linear matrix space, and we use the reduced row echelon form of ω when viewed as a matrix in $\mathbb{F}_q^{d \times n^2}$ as the complete universe invariant. This satisfies condition 2 of Theorem 3.23. The action is computable in polynomial time (second part of condition 1) as it only involves inverting and multiplying matrices in $\mathrm{GL}_n(\mathbb{F}_q)$.

The remaining conditions only depend on the underlying group, which is different from before, namely $\mathrm{GL}_n(\mathbb{F}_q)$ instead of S_n. Products and inverses in $\mathrm{GL}_n(\mathbb{F}_q)$ can be computed in polynomial time (condition 4), and the identity mapping serves as the complete group invariant (condition 6). Thus, only the uniform sampler for $\mathrm{GL}_n(\mathbb{F}_q)$ (first part of condition 1) and the pac overestimator for $|\mathrm{GL}_n(\mathbb{F}_q)|$ (condition 5) remain to be argued.

The standard way of constructing the elements of $\mathrm{GL}_n(\mathbb{F}_q)$ consists of n steps, where the i-th step picks the i-th row as any row vector that is linearly independent of the $(i-1)$ prior ones. The number of choices in the i-th step is $q^n - q^{i-1}$. Thus, $|\mathrm{GL}_n(\mathbb{F}_q)| = \prod_{i=1}^{n}(q^n - q^{i-1})$, which can be computed in time $\mathrm{poly}(|x|)$ (condition 5). It also follows that the probability that a random $(n \times n)$-matrix over $\mathbb{F}_q$ is in $\mathrm{GL}_n(\mathbb{F}_q)$ is at least some positive constant (independent of n and q), which implies that $\{H_x\}$ can be uniformly sampled in time $\mathrm{poly}(|x|)$, satisfying the first part of condition 1.

Corollary 3.28. *Matrix Subspace Conjugacy is in* $\mathrm{ZPP}^{\mathrm{MKTP}}$.

Before closing, we note that there is an equivalent of the Lehmer code for $\mathrm{GL}_n(\mathbb{F}_q)$. We do not need it for our results, but it may be of interest in other contexts. In general, Lehmer's approach works for indexing objects that consist of multiple components where the set of possible values for the i-th component may depend on the values of the prior components, but the *number* of possible values for the i-th component is independent of the values of the prior components. An efficiently decodable indexing follows provided one can efficiently index the possible values for the i-th component given the values of the prior components. The latter is possible for $\mathrm{GL}_n(\mathbb{F}_q)$. We include a proof for completeness.

Proposition 3.29. *For each n and prime power q,* $\mathrm{GL}_n(\mathbb{F}_q)$ *has an indexing that is uniformly decodable in time* $\mathrm{poly}(n, \log(q))$.

Proof. Consider the above process. In the i-th step, we need to index the complement of the subspace spanned by the $i-1$ row vectors picked thus far, which are linearly independent. This can be done by extending those $i-1$ row vectors by $n-i+1$ new row vectors to a full

basis, and considering all q^{i-1} linear combinations of the $i-1$ row vectors already picked, and all $(q^{n-i+1}-1)$ *non-zero* linear combinations of the other basis vectors, and outputting the sum of the two components. More precisely, on input $k \in [q^n - q^{i-1}]$, write $k-1$ as $k_0 + k_1 q^{i-1}$ where k_0 and k_1 are nonnegative integers with $k_0 < q^{i-1}$, and output $v_0 + v_1$ where v_0 is the combination of the $i-1$ row vectors already picked with coefficients given by the binary expansion of k_0, and v_1 is linear combination of the other basis vectors with coefficients given by the binary expansion of $k_1 + 1$. Using Gaussian elimination to construct the other basis vectors, the process runs in time $\mathrm{poly}(n, \log(q))$. ∎

3.7 Coset Indexings and Normal Forms for Permutation Groups

In this section we develop the efficiently decodable indexings for cosets of permutation subgroups claimed in Lemma 3.9, and also use some of the underlying ideas to establish the normal form for permutation groups stated in Lemma 3.26.

Indexing Cosets The indexings are not strictly needed for our main results as the generic encoding from the Encoding Lemma can be used as a substitute. However, the information-theoretic optimality of the indexings may be useful in other contexts. In fact, we present a further generalization that may be of independent interest, namely an efficiently decodable indexing for cosets of permutation subgroups within another permutation subgroup.

Lemma 3.30. *For all $\Gamma \leq H \leq S_n$, there exists an indexing of the cosets of Γ within H that is uniformly decodable in polynomial time when Γ and H are given by a list of generators.*

Lemma 3.9 is just the instantiation of Lemma 3.30 with $H = S_n$.

In the following, we arbitrarily work with left ($\pi\Gamma$) as opposed to right ($\Gamma\pi$) cosets; all our results hold for both, however, as one can switch from one statement to the other by taking inverses. Related to this, there is an ambiguity regarding the order of application

in the composition gh of two permutations: first apply g and then h, or vice versa. Both interpretations are fine. For concreteness, we assume the former.

The proof of Lemma 3.30 requires some elements of the theory of permutation groups. Given a list of permutations $\pi_1, \ldots, \pi_k \in S_n$, we write $\Gamma = \langle \pi_1, \ldots, \pi_k \rangle \leq S_n$ for the subgroup they generate. Given a permutation group $\Gamma \leq S_n$ and a point $i \in [n]$, the Γ-orbit of i is the set $\{g(i) : g \in \Gamma\}$, and the Γ-stabilizer of i is the subgroup $\{g \in \Gamma : g(i) = i\} \leq \Gamma$.

We make use of the fact that (a) the number of cosets of a subgroup Γ of a group H equals $|H|/|\Gamma|$, and (b) the orbits of a subgroup Γ of H form a refinement of the orbits of H. We also need the following basic routines from computational group theory (see, for example, [HEO05; Ser03]).

Proposition 3.31. *Given a set of permutations that generate a subgroup $\Gamma \leq S_n$, the following can be computed in time polynomial in n:*

(1) the cardinality $|\Gamma|$,

(2) a permutation in Γ that maps u to v for given $u, v \in [n]$, or report that no such permutation exists in Γ, and

(3) a list of generators for the subgroup Γ_v of Γ that stabilizes a given element $v \in [n]$.

The proof of Lemma 3.30 makes implicit use of an efficient process for finding a *canonical representative* of $\pi\Gamma$ for a given permutation $\pi \in H$, where "canonical" means that the representative depends on the coset $\pi\Gamma$ only. The particular canonical representative the process produces can be specified as follows.

Definition 3.32. For a permutation $\pi \in S_n$ and a subgroup $\Gamma \leq S_n$, the *canonical representative* of π modulo Γ, denoted $\pi \bmod \Gamma$, is the least $\pi' \in \pi\Gamma$, with respect to the lexicographic order of the sequence $(\pi'(1), \pi'(2), \ldots, \pi'(n))$. ◂

The process is well-known. We spell it out in the proof of the following lemma as it provides intuition for the proof of Lemma 3.30.

Lemma 3.33 (Theorem 10 in [AK06]). *There exists a polynomial-time algorithm that takes as input a generating set for a subgroup $\Gamma \leq S_n$ and a permutation $\pi \in S_n$, and outputs the canonical representative $\pi \bmod \Gamma$.*

Proof of Lemma 3.33. Consider the element 1 of $[n]$. Permutations in $\pi\Gamma$ map 1 to an element v in the same Γ-orbit as $\pi(1)$, and for every element v in the Γ-orbit of $\pi(1)$ there exists a permutation in $\pi\Gamma$ that maps 1 to v. We can canonize the behavior of π on the element 1 by replacing π with a permutation $\pi_1 \in \pi\Gamma$ that maps 1 to the minimum element m in the Γ-orbit of $\pi(1)$. This can be achieved by multiplying π to the right with a permutation in Γ that maps $\pi(1)$ to m.

Next we apply the same process to π_1 but consider the behavior on the element 2 of $[n]$. Since we are no longer allowed to change the value of $\pi_1(1)$, which equals m, the canonization of the behavior on 2 can only use multiplication on the right with permutations in Γ_m, i.e., permutations in Γ that stabilize the element m. Doing so results in a permutation $\pi_2 \in \pi_1\Gamma$.

We repeat this process for all elements $k \in [n]$ in order. In the k-th step, we canonize the behavior on the element k by multiplying on the right with permutations in $\Gamma_{\pi_{k-1}([k-1])}$, i.e., permutations in Γ that pointwise stabilize all of the elements $\pi_{k-1}(\ell)$ for $\ell \in [k-1]$. ■

Proof of Lemma 3.30. The number of canonical representatives modulo Γ in H equals the number of distinct (left) cosets of Γ in H, which is $|H|/|\Gamma|$. We construct an algorithm that takes as input a list of generators for Γ and H, and an index $i \in [|H|/|\Gamma|]$, and outputs the permutation σ that is the lexicographically i-th canonical representative modulo Γ in H.

The algorithm uses a prefix search to construct σ. In the k-th step, it knows the prefix $(\sigma(1), \sigma(2), \ldots, \sigma(k-1))$ of length $k-1$, and needs to figure out the correct value $v \in [n]$ to extend the prefix with. In order to do so, the algorithm needs to compute for each $v \in [n]$ the count c_v of canonical representatives modulo Γ in H that agree with σ on $[k-1]$ and take the value v at k. The following claims allow us to do that efficiently when given a permutation $\sigma_{k-1} \in H$ that agrees with σ on $[k-1]$. The claims use the notation $T_{k-1} \doteq \sigma_{k-1}([k-1])$,

which also equals $\sigma([k-1])$.

Claim 3.34. *The canonical representatives modulo Γ in H that agree with $\sigma \in H$ on $[k-1]$ are exactly the canonical representatives modulo $\Gamma_{T_{k-1}}$ in $\sigma_{k-1}H_{T_{k-1}}$.*

Proof. The following two observations imply Claim 3.34.

(i) A permutation $\pi \in H$ agrees with $\sigma \in H$ on $[k-1]$
$\Leftrightarrow$ π agrees with σ_{k-1} on $[k-1]$
$\Leftrightarrow$ $\sigma_{k-1}^{-1}\pi \in H_{T_{k-1}}$
$\Leftrightarrow$ $\pi \in \sigma_{k-1}H_{T_{k-1}}$.

(ii) Two permutations in $\sigma_{k-1}H_{T_{k-1}}$, say $\pi \doteq \sigma_{k-1}g$ and $\pi' \doteq \sigma_{k-1}g'$ for $g, g' \in H_{T_{k-1}}$, belong to the same left coset of Γ iff they belong to the same left coset of $\Gamma_{T_{k-1}}$. This follows because if $\sigma_{k-1}g' = \sigma_{k-1}gh$ for some $h \in \Gamma$, then h equals $g^{-1}g' \in H_{T_{k-1}}$, so $h \in \Gamma \cap H_{T_{k-1}} = \Gamma_{T_{k-1}}$.

■

Claim 3.35. *The count c_v for $v \in [n]$ is nonzero iff v is the minimum of some $\Gamma_{T_{k-1}}$-orbit contained in the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$.*

Proof. The set of values of $\pi(k)$ when π ranges over $\sigma_{k-1}H_{T_{k-1}}$ is the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$. Since $\Gamma_{T_{k-1}}$ is a subgroup of $H_{T_{k-1}}$, this orbit is the union of some $\Gamma_{T_{k-1}}$-orbits. Combined with Claim 3.34 and the construction of the canonical representatives modulo $\Gamma_{T_{k-1}}$, this implies Claim 3.35. ■

Claim 3.36. *If a count c_v is nonzero then it equals $|H_{T_{k-1}\cup\{v\}}|/|\Gamma_{T_{k-1}\cup\{v\}}|$.*

Proof. Since the count is nonzero, there exists a permutation $\sigma' \in H$ that is a canonical representative modulo Γ that agrees with σ_{k-1} on $[k-1]$ and satisfies $\sigma'(k) = v$. Applying Claim 3.34 with σ replaced by σ', k by $k' \doteq k+1$, T_{k-1} by $T'_k \doteq T_{k-1} \cup \{v\}$, and σ_{k-1} by any permutation $\sigma'_k \in H$ that agrees with σ' on $[k]$, yields Claim 3.36. This is because the

Algorithm 1

Input: positive integer n, $\Gamma \leq H \leq S_n$, $i \in [|H|/|\Gamma|]$
Output: lexicographically i-th canonical representative modulo Γ in H

$\sigma_0 \leftarrow id$
for $k = 1$ to n **do**
    $O_1, O_2, \ldots \leftarrow$ Γ-orbits contained in the H-orbit of $\sigma_{k-1}(k)$, in increasing order of $\min(O_i)$
    find integer ℓ such that $\sum_{j=1}^{\ell-1} c_{\min(O_j)} < i \leq \sum_{j=1}^{\ell} c_{\min(O_j)}$, where $c_v \doteq |H_v|/|\Gamma_v|$
    $i \leftarrow i - \sum_{i=1}^{\ell-1} c_{\min(O_j)}$
    $m \leftarrow \min(O_\ell)$
    find $\tau \in H$ such that $\tau(\sigma_{k-1}(k)) = m$
    $\sigma_k \leftarrow \sigma_{k-1}\tau$
    $H \leftarrow H_m$; $\Gamma \leftarrow \Gamma_m$
return σ_n

number of canonical representatives modulo $\Gamma_{T'_k}$ in $\sigma'_k H_{T'_k}$ equals the number of (left) cosets of $\Gamma_{T'_k}$ in $H_{T'_k}$, which is the quantity stated in Claim 3.36. ∎

The algorithm builds a sequence of permutations $\sigma_0, \sigma_1, \ldots, \sigma_n \in H$ such that σ_k agrees with σ on $[k]$. It starts with the identity permutation $\sigma_0 = id$, builds σ_k out of σ_{k-1} for increasing values of $k \in [n]$, and outputs the permutation $\sigma_n = \sigma$.

Pseudocode for the algorithm is presented in Algorithm 1. Note that the pseudocode modifies the arguments Γ, H, and i along the way. Whenever a group is referenced in the pseudocode, the actual reference is to a list of generators for that group.

The correctness of the algorithm follows from Claims 3.35 and 3.36. The fact that the algorithm runs in polynomial time follows from Proposition 3.31. ∎

Normal Form Finally, the canonization captured in Definition 3.32 and Lemma 3.33 is used to establish the normal form for permutation groups given by Lemma 3.26 (restated below):

Lemma 3.37 (Lemma 3.26, restated). *There is a polynomial-time algorithm ν that takes as input a list L of elements of S_n, and outputs a list of generators for the subgroup generated*

by the elements in L such that for any two lists L_0, L_1 of elements of S_n that generate the same subgroup, $\nu(L_0) = \nu(L_1)$.

Proof. Let Γ denote the subgroup generated by L, and recall that $\Gamma_{[i]}$ denotes the subgroup of Γ that stabilizes each element in $[i]$, for $i \in \{0, 1, \ldots, n\}$. We have that $\Gamma_{[0]} = \Gamma$, and $\Gamma_{[n-1]}$ consists of the identity only.

We define $\nu(L)$ as follows. Start with ν being the empty list. For $i \in [n-1]$, in the i-th step we consider each $j \in [n]$ that is in the $\Gamma_{[i-1]}$-orbit of i in order. Note that for each such j, the permutations in $\Gamma_{[i-1]}$ that map i to j form a coset of $\Gamma_{[i-1]} \bmod \Gamma_{[i]}$. We append the canonical representative of this coset to ν. $\nu(L)$ is the value of ν after step $n-1$.

As we only include permutations from Γ, $\nu(L)$ generates a subgroup of Γ. By construction, for each $i \in [n-1]$, the permutations we add in the i-th step represent all cosets of $\Gamma_{[i-1]} \bmod \Gamma_{[i]}$. It follows by induction on $n-i$ that the permutations added to ν during and after the i-th step generate $\Gamma_{[i-1]}$ for $i \in [n]$. Thus, $\nu(L)$ generates $\Gamma_{[0]} = \Gamma$.

That $\nu(L)$ only depends on the subgroup Γ generated by L follows from its definition, which only refers to the abstract groups $\Gamma_{[i]}$, their cosets, and their canonical representatives. That $\nu(L)$ can be computed in polynomial time follows by tracking a set of generators for the subgroups $\Gamma_{[i]}$ based on Proposition 3.31. More specifically, we use item 2 to check whether a given j is in the $\Gamma_{[i-1]}$-orbit of i, and item 3 to obtain $\Gamma_{[i]}$ out of $\Gamma_{[i-1]}$ as $\Gamma_{[i]} = (\Gamma_{[i-1]})_i$. ■

Chapter 4

Inversion Minimization

4.1 Overview

In this chapter, we present some theory developed for the *comparison-query* model of computation. In this model, there is a set of n *items*. The input is an unknown *ranking* of the items: one item is "first" (rank 1), another is "second" (rank 2), and so on, until the final item which is "last" (rank n). Initially we know nothing about the ranking, and we want to solve some problem that requires us to uncover some information about the ranking. We may learn about the ranking through queries with a particular form: select two distinct items, i and j, and ask, "how does the rank of i compare to the rank of j?". The response is either "the rank of i is less than the rank of j" or "the rank of i is more than the rank of j". By asking enough queries, we can distinguish any one ranking from any other; thus any function from rankings to some codomain can be computed in this model. The question is, for each problem of interest, how many queries are necessary and sufficient to solve it?

For some problems, we have good answers to this question.

- The canonical example is *sorting*, which is tantamount to outputting the entire input ranking. Standard algorithms such as mergesort and heapsort give an upper bound of $\log_2(n!) + O(n)$ queries. This has been improved to $\log_2(n!) + o(n)$ [Ser21]. The

folklore lower bound says that it is necessary to make at least $\log_2(n!)$ queries, because each of the $n!$ input rankings requires a distinct execution trace, each execution trace is determined by the results of its queries, and each query has only two possible outcomes.

- In *selection* we are told a rank r, and then must identify the item with rank r. It is known that $\Theta(n)$ comparisons are necessary and sufficient for selection [BFP+73; DZ99; DZ01].

  There is also *multiple selection* in which one is given multiple ranks $r_1, r_2, \ldots, r_k$, and must identify the item with rank r_1, the item with rank r_2, etc, up to the item with rank r_k. The complexity of multiple selection is likewise known up to a $\Theta(n)$ gap between the upper and lower bounds [KMM+05].

- In *min-heap construction*, we must arrange the items as nodes in a complete binary tree such that every node besides the root is greater than its parent. It is known that $\Theta(n)$ comparisons are necessary and sufficient for min-heap construction. Max-heap construction is symmetric.

All the problems above can be cast as instantiations of a general framework, known as *Partial Order Production*, introduced by Schönhage [Sch76]. Here, in addition to the unknown ranking of the items, we are given n *slots* as well as a known partial order on the slots, $<_{\mathtt{s}}$. The objective of partial order production is to put each item into a slot, one item per slot, so that whenever two slots s_1, s_2 are related by $s_1 <_{\mathtt{s}} s_2$, the item in s_1 is ranked less than the item in s_2.

Sorting coincides with the case where $<_{\mathtt{s}}$ is a total order. In selection of rank r, there is a designated slot s, and there are exactly $r-1$ slots s' with $s' <_{\mathtt{s}} s$ and exactly $n-r$ slots s' with $s' >_{\mathtt{s}} s$; there are no other relations in $<_{\mathtt{s}}$. Multiple selection is similar. For heap construction, $<_{\mathtt{s}}$ matches the complete binary tree arrangement.

There is a generic lower bound for partial order production, the information-theoretic limit. For each way of putting items into slots, the number of input rankings for which that

way is a correct answer is bounded by $e(<_s)$, the number of ways to extend $<_s$ to a total order. Therefore there must be at least $n!/e(<_s)$ distinct execution traces. Since each execution trace is determined by the results of its queries, and each query has only two outcomes, we conclude that $L(<_s) \doteq \log_2(n!/e(<_s))$ queries are necessary to solve partial order production.

Complementing this lower bound is an upper bound of $(1 + o(1)) \cdot L(<_s) + O(n)$ queries [CFJ+10]. One may assume without loss of generality the relationship $L(<_s) \geq n-1$, in which case $O(L(<_s))$ queries always suffices. Thus the complexity of partial order production is $\Theta(L)$.

Not every problem of interest is an instance of partial order production. Here are a few examples.

- In *rank finding*, there is a designated item, and we have to compute its rank. For this it is necessary and sufficient to perform $n - 1$ comparisons.
- In *counting inversions*, the items are arranged in some known order, independent of the input ranking, and the objective is to count the number of *inversions* between that order and the input ranking. An inversion is a pair of items such that their order in the arrangement is opposite the order of their ranks. As we discuss later, counting inversions has exactly the same complexity as sorting.
- The problem of *inversion parity* is the same as counting inversions except that one need only count the number of inversions modulo 2. This problem, as well as variations like counting inversions modulo m for any $m > 1$, likewise turns out to have exactly the same complexity as sorting.

For each of the problems above, information theory does not provide a satisfactory lower bound. For example, in the inversion parity problem there are only two possible answers, so information theory only gives a lower bound of one query, whereas the right complexity is $\Theta(n \log n)$. It so happens that for each of the preceding three examples, the query complexity is known quite precisely; however, in each case the arguments are quite problem-specific.

Minimum Inversions in a Tree A less well-understood problem is *minimum inversions in a tree*. Following how the problem appeared in an article by Degerman in the psychology literature [Deg82], we imagine that a clustering analysis has been done on n items, yielding a binary tree with the items at the leaves, and we want to visualize the tree. The usual way to visualize a tree requires that we first choose an *arrangement* for it, i.e., for every internal node of the tree, what shall be the left-to-right order of its children. As an example, consider $n = 3$ items, `A`, `B`, and `C`, that have been clustered such that `A` is first merged with `B` and then they are merged with `C`. There are four possible arrangements:

In general there are many ways to arrange the tree, and the choice can affect the utility of the visualization. The question thus becomes how to choose the best arrangement. In the presence of some external ranking of the items, Degerman proposes to minimize the number of inversions between the order of the items in the arrangement and in the external ranking. Among other benefits, note that the minimum number of inversions behaves as a measure of correlation between the external ranking and the clustering: the smaller the number of inversions, the more correlated the ranking and the clustering are.

There are a number of variations on this problem according to whether one wants to find an optimal arrangement, the minimum number of inversions, or both, and according to whether one wants to allow for more general tree shapes than just binary. When general tree shapes are allowed, finding the optimal arrangement can be as hard as sorting. Lower bounds for computing only the minimum number of inversions, however, is less clear. Thus for the purposes of this chapter, we consider the following problem.

Definition 4.1. The *minimum inversions in a tree* problem is the following. The input consists of a rooted tree T and a ranking of the leaves of T, and the output is the minimum number of inversions between some arrangement of T and the input ranking. ◂

Here is a nontrivial example.

Example 4.2. Consider the tree whose root has two children, one of which is a leaf, and the other of which has the other $n-1$ leaves as its children. Here are two relevant arrangements of the tree, where the label of each leaf is its rank in the input ranking:

Arranging the tree amounts to deciding whether to put the lone leaf before or after all the other leaves, and then deciding how to order the others. When the rank of the lone leaf is r, the number of inversions between the lone leaf and the others is $r-1$ (if placed before the others) or $n-r$ (if placed after). By sorting the other leaves, we can ensure that there are zero inversions among them. The answer is thus $\min(r-1, n-r)$. We can compute this in $n-1$ comparisons by comparing the lone leaf to all the other leaves, which determines r. ◂

One way to solve minimum inversions in a tree is just to learn the entire input ranking via a sorting algorithm. This gives an upper bound of $\log_2(n!) + o(n)$ comparisons [Ser21]. As Example 4.2 shows, better algorithms can exist for specific tree shapes, but it is not clear whether it is possible to improve over sorting in the general case.

Lower bounds, on the other hand, seem to be entirely unexplored. Information theory does not give a convincing bound: since the number of inversions is no more than $\binom{n}{2}$, the best lower bound one can get is $\log_2 \binom{n}{2} = 2\log_2(n) - O(1)$. Moreover, as we see in Example 4.2, it is possible in some cases to do significantly better than $\log_2(n!)$ comparisons. Thus we cannot hope that in general the problem will turn out to be equivalent to sorting, as is the case for counting inversions and inversion parity.

Lower Bound Framework Toward proving lower bounds for the minimum inversions in a tree problem, we formulate a general-purpose framework for proving lower bounds in

the comparison model. First of all, we use the standard comparison-based decision trees to formalize algorithms in the model. We will be interested in lower bounds for the height of decision trees that compute various functions from the set of rankings of a fixed set of items to some codomain. Our lower bounds for the height always come by proving lower bounds on the number of leaves, the binary logarithm of which is a lower bound for the height.

The conceptual fulcrum in our framework is a form of certificate complexity suited to the comparison model. Fix a decision tree and a leaf ℓ in the tree, and let Q denote the set of queries and their results along the path from the root to ℓ. Every input ranking that is consistent with Q of course follows this root-to-leaf path and winds up at ℓ, whence the output is some y. In short, consistency of the input with Q implies the output is y. In this manner, we say that Q *certifies* that the output is y.

Since the rankings are all orders, we can equivalently convert Q into a partial order $<_{\mathtt{q}}$ such that the consistency of a ranking with Q is equivalent to the ranking being a linear extension of $<_{\mathtt{q}}$. That is, the set of rankings that reach ℓ is precisely the set of rankings that extend $<_{\mathtt{q}}$. As we vary ℓ, the sets of rankings that reach ℓ form a partition of the set of all rankings. As we just discussed, this partition satisfies two properties: first, every part of this partition equals the set of linear extensions of some partial order; second, for every part of the partition, the algorithm produces the same output. Since moreover the number of parts in the partition is at most (and without loss of generality equal to) the number of leaves in the decision tree, we are led to the following complexity measure.

Definition 4.3. Let I be a set of items. For any function Π from rankings of I to some codomain, the *partial-order certificate complexity (p.o.-certificate complexity)* of Π is the fewest number of parts in any partition Λ of the rankings of I such that

- Π is constant on each part of Λ, and
- for every part R of Λ, there is a partial order $<_{\mathtt{q}}$ of I such that R equals the set of rankings that extend $<_{\mathtt{q}}$.

◂

For any problem, the partial-order certificate complexity is a lower bound on the number of leaves in any decision tree that solves the problem. However, the partial-order certificate complexity can be much lower than the minimum number of leaves in a decision tree solving the problem; see Example 4.20 in Section 4.4.

From here, it suffices to prove lower bounds on the partial-order certificate complexity of problems of interest. We present two approaches to doing so. Both proceed by considering the effect on the output of perturbations to the input that are hard for queries to observe. More specifically, we consider the following perturbations:

Definition 4.4. An *adjacent-rank transposition* is a permutation σ of the set $[n]$ of ranks of the form $\sigma = (r\ \ s)$ with $|r - s| = 1$. ◂

As with any permutation of the set of ranks, adjacent-rank transpositions act on the rankings of the items by changing the rank of each item according to the permutation. For example, (2 3) swaps the order of the items that have ranks 2 and 3. Adjacent-rank transpositions are the least noticeable perturbations one can apply to a ranking in the following sense: if two rankings differ by an adjacent-rank transposition, then the only query that distinguishes them is the query that compares the two items with the affected ranks.

Connectivity We start with an approach that looks at connectivity in the graph with rankings as vertices and the adjacent-rank transpositions as edges. The following lemma states that each partition class of Λ in Definition 4.3 induces a *connected* subgraph of this graph.

Lemma 4.5 (Connectivity Lemma). *Let I be a set of items, and let $<_{\mathtt{q}}$ be a partial order of the items. For any two rankings that extend $<_{\mathtt{q}}$, there exists a sequence of adjacent-rank*

transpositions taking one ranking to the other such that every intermediate rankings is also consistent with $<_q$.

One particular consequence of the Connectivity Lemma is that whenever an algorithm fails to distinguish a pair of rankings, it fails to distinguish a pair of rankings that differ by an adjacent-rank transposition. This allows for a simple exposition of the lower bound for counting inversions as well as inversion parity, as follows. Every adjacent-rank transposition changes the number of inversions by exactly one (up or down), so every algorithm must distinguish every pair of rankings that differ by an adjacent-rank transposition. By the Connectivity Lemma, this means the algorithm must in fact distinguish *every* pair of rankings. Meanwhile, distinguishing every pair of rankings is equivalently sorting the input. This and additional consequences of the Connectivity Lemma are discussed in Section 4.4.

One other consequence of the Connectivity Lemma is a lower bound for special cases of minimizing inversions in trees. In the particular case of a perfect binary tree, the required number of comparisons is at least $\log_2(n!) - O(n)$ (Corollary 4.46). The argument relies on fragile parity conditions that happen to hold in the case of perfect binary trees but breaks down for more general tree shapes. We present a more robust argument with our next tool.

Sensitivity We adapt the complexity measure *sensitivity* from Boolean query complexity to the comparison model. Broadly, a problem is sensitive if many ways of perturbing the input cause a change in the output. Depending on how the input and perturbation are quantified, one gets different sensitivity measures. The common intuition in query complexity is that, as long as the perturbations are hard for queries to notice, sensitive problems will require many queries.

We specifically study the *average sensitivity to adjacent-rank transpositions*. That is, the input ranking is a uniformly random ranking, and we consider perturbing it by adjacent-rank transpositions.

Definition 4.6. Let I be a set of items, and let Π be any function from the rankings of I to some codomain. For a fixed input ranking $<_i$ and adjacent-rank transposition σ, we say that Π is *sensitive* to σ at $<_i$ if $\Pi(<_i) \neq \Pi(\sigma(<_i))$. The *average sensitivity* of Π, denoted $s(\Pi)$, is the average number of adjacent-rank transpositions σ such that Π is sensitive to σ at $<_i$, when $<_i$ is drawn uniformly from all rankings of I. ◂

Sorting, counting inversions, and inversion parity have full sensitivity $n-1$ at every input ranking $<_i$ and thus achieve the maximum average sensitivity of $n-1$. In contrast, selection of rank 1 or n has sensitivity 1 at every input ranking $<_i$, so the overall average sensitivity is 1. For selection of other ranks, the sensitivity is 2 at every input ranking $<_i$, so the overall average sensitivity is 2.

We show that if the average sensitivity of a problem is high, then the partial-order certificate complexity of the problem must be high, and therefore the problem requires many queries to compute.

Lemma 4.7 (Sensitivity Lemma). *Let I be a set of n items, and let $\Phi : [1, n] \to \mathbb{R}$ be any nondecreasing convex function with $\Phi(k) = k!$ for $k = 1, \ldots, n$. For any function Π from the rankings of I to some codomain, every comparison-based decision tree that computes Π must have at least*

$$\frac{1}{n}\Phi(1 + s(\Pi)) \tag{4.1}$$

leaves, and hence have height at least

$$\log_2 \Phi(1 + s(\Pi)) - \log_2 n. \tag{4.2}$$

For sorting, counting inversions, and inversion parity, this recovers the standard lower bound up to a small loss. For selection, the bound is not so good, reflecting that the average sensitivity is not always capable of proving strong lower bounds. However, a substantial advantage of going through sensitivity is that it is can often be easier to estimate the average sensitivity than it is to argue a query lower bound from scratch.

Using the Sensitivity Lemma, we prove lower bounds on the complexity of minimum inversions in a tree. We show that the sensitivity is unconditionally at least $n/3$ in binary trees, which by the Sensitivity Lemma entails an $\Omega(n \log n)$ lower bound for the problem. Assuming moreover a conjecture on the so-called Gaussian binomial coefficients, we show that the sensitivity in the case of binary trees is in fact $n - O(1)$, and this entails a lower bound of $\log_2(n!) - O(\log n)$. The conjecture is known to be true in some special cases, and in particular the improved bound holds unconditionally in the case of a perfect binary tree. Our results extend beyond binary trees as well.

Organization We formally introduce the comparison model in Section 4.2. Section 4.3 presents the lower bound framework, and then Sections 4.4 and 4.5 develop the Connectivity Lemma and the Sensitivity Lemma, respectively. We give a first analysis of the sensitivity of the minimum inversions in a tree problem in Section 4.6, and then refine the analysis in Section 4.7.

4.2 Comparison-Query Model of Computation

In this section we formally describe the standard comparison-query model of computation. As a query model, computational state can be understood in terms of what the computer "knows" at various points in time.

Initially, there is some *context* of information that the computer knows. In the comparison query model, the context always includes (but is not limited to) a finite set, I. The elements of I are referred to as the *items*, and the size of I is denoted by n. What the computer does not know is its *input*. In the comparison model, the input is always just a total order on I. We denote the input order with the subscript $\mathtt{i}$, e.g., $<_{\mathtt{i}}$. In some problems, such as selection, there is additional information that is part of the "input", such as the rank of the item to select. We treat this information as part of the context.

As time progresses, the computation evolves by *querying* the input and thus growing its knowledge about it. In the comparison model, queries take the following form.

Definition 4.8. Let I be a set of items. A *comparison query* (or query for short) is any ordered pair (i, j) of two distinct items $i, j \in I$.

Relative to a total order $<_{\mathtt{i}}$ on I, every query has a *result*. If $i <_{\mathtt{i}} j$, then the result is "<"; otherwise, $i >_{\mathtt{i}} j$, and the result is ">". ◂

With each query, a computation learns the result of the query relative to the hidden input and can use the result (and all past results) to determine the next query that it makes. Eventually, the computation halts and produces an output, which can likewise depend on all preceding results. Formally, the model is a decision tree.

Definition 4.9. Let I be a set of items. A *comparison-based decision tree* with queries in I is a rooted tree extended with the following data. Every non-leaf vertex is labeled by a comparison query with items in I and has exactly two children, one for "<" and one for ">". Every leaf has a piece of arbitrary data called its *output label*. ◂

Every comparison-based decision tree $\mathcal{T}$ computes a function from the set of total orders of I to the set of output labels of its leaves. Given a total order $<_{\mathtt{i}}$ as input, one starts at the root and follows the results of the queries until one arrives at a leaf, whence $\mathcal{T}(<_{\mathtt{i}})$ is the output label of that leaf.

We are interested in understanding the performance of comparison-based decision trees at solving various problems of interest. One can formalize a problem as a class Ξ of contexts, where for each context $\xi \in \Xi$, there is a set of items I_ξ, a set of outputs $\mathcal{O}_\xi$, and a function Π_ξ from the total orders of I_ξ to $\mathcal{O}_\xi$. To solve a problem, one provides for every $\xi \in \Xi$ a comparison-based decision tree $\mathcal{T}_\xi$ with queries in I_ξ such that $\mathcal{T}_\xi$ computes Π_ξ.

Example 4.10. We demonstrate the above formalism in the case of sorting. A context for sorting consists of a set I_ξ of n items as well as a totally-ordered set S_ξ of n *slots*. We denote

the order on S_ξ by $<_{\mathtt{s}}$. The set of outputs is the set of bijections $f : I_\xi \to S_\xi$. For each input total order $<_{\mathtt{i}}$, $\Pi_\xi(<_{\mathtt{i}})$ is the bijection f such that $f(i) <_{\mathtt{s}} f(j)$ holds whenever $i <_{\mathtt{i}} j$. ◂

For our purposes, we will generally fix the context and suppress notation that refers to it. Thus we fix the set I of items and consider a "problem" as just a single function Π from the total orders of I to some codomain. To solve a problem, one provides a single decision tree $\mathcal{T}$ with queries in I that computes Π.

In this chapter, we will measure the performance of a comparison-based decision tree $\mathcal{T}$ in two ways: the height of $\mathcal{T}$, denoted $\mathrm{ht}(\mathcal{T})$, and the number of leaves in $\mathcal{T}$, denoted $\mathrm{L}(\mathcal{T})$. The height captures the worst-case number of queries that are performed over all inputs. The number of leaves counts the number of possible executions and is useful as a proxy toward lower bounds on the height. In particular, all our lower bounds on $\mathrm{ht}(\mathcal{T})$ follow from the relationship

$$\mathrm{ht}(\mathcal{T}) \geq \log_2 \mathrm{L}(\mathcal{T}). \tag{4.3}$$

4.3 Lower Bound Framework

In this section, we introduce the framework for our lower bounds. For the remainder of the section, fix a set I of n items.

We begin with a shift in perspective. Rather than use the relational $<_{\mathtt{i}}$ notation for the input order, we change to a notation that makes the ranks of the items more explicit.

Definition 4.11. Let I be a set of n items. A *ranking* is any bijection $\rho : I \to [n]$. ◂

We refer to the elements of the codomain $[n]$ as *ranks*; a ranking is thus an assignment of a unique rank to every position. The ranks are always taken to have their usual total order $1 < 2 < \cdots < n$. Every ranking ρ encodes a total order $<_{\mathtt{i}}$ of the items according to $i <_{\mathtt{i}} j$ if and only if $\rho(i) < \rho(j)$. For every total order of the items, there is exactly one ranking that encodes it. Thus the set of total orders of the items is equivalently the set of rankings of the items.

Definition 4.12. For a set of n items I, define $\mathcal{R}(I)$ to be the set of rankings $\rho : I \to [n]$. ◂

Next, we note that a critical limitation of decision trees is that any two inputs that lead to the same leaf of the tree must have the same output. That is, the decision tree cannot distinguish the two inputs. This leads to the following notion.

Definition 4.13. Let I be a set of items, and let $\mathcal{T}$ be a comparison-based decision tree with queries in I. The *indistinguishability partition* of $\mathcal{T}$, denoted $\Lambda_{\mathcal{T}}$, is the partition of $\mathcal{R}(I)$ where two rankings are in the same part of the partition if and only if they lead to the same leaf of $\mathcal{T}$. ◂

The function computed by $\mathcal{T}$ is necessarily constant on every part of $\Lambda_{\mathcal{T}}$. We use the terminology *part-wise constant* to refer to such a function. The indistinguishability partition in fact completely captures the complexity of $\mathcal{T}$. This is because the output labels of $\mathcal{T}$ can be changed at no cost, and thus $\mathcal{T}$ can be adapted to compute any function that is part-wise constant on $\Lambda_{\mathcal{T}}$.

The following elementary relationships connect the complexity measures of $\mathcal{T}$ to its indistinguishability partition:

Fact 4.14. *Let $\mathcal{T}$ be a comparison based decision tree, and let $\Lambda_{\mathcal{T}}$ be its leaf partition. The following relationships hold:*

$$\mathrm{L}(\mathcal{T}) \geq \#\Lambda_{\mathcal{T}} \tag{4.4}$$

$$\mathrm{ht}(\mathcal{T}) \geq \log_2 \#\Lambda_{\mathcal{T}} \tag{4.5}$$

where $\#\Lambda_{\mathcal{T}}$ is the number of parts in $\Lambda_{\mathcal{T}}$.

Proof. The first inequality follows from the definition of $\Lambda_{\mathcal{T}}$, and the second follows from (4.3). ■

Now we relate the structure of $\Lambda_{\mathcal{T}}$ to the computation in $\mathcal{T}$. Let R be one of the parts of $\Lambda_{\mathcal{T}}$. Every ranking in R follows the same root-to-leaf path through $\mathcal{T}$. Along that path

is a set of queries, and for all of those queries, every ranking in R has the same result. Let Q be the set of pairs (i, j) such that either (i, j) is a query along that path and its result is "<", or else (j, i) is a query along that path and its result is ">". By construction, every ranking ρ in R has $\rho(i) < \rho(j)$ for every $(i, j) \in Q$. Moreover, R contains *every* ranking with that property.

From here, we close Q under "logical inference", in the sense that we add to Q any pairs (i, j) such that $\rho(i) < \rho(j)$ for every $\rho \in R$. The resulting set of pairs is equivalently the partial order $<_{\mathtt{q}}$ generated by $i <_{\mathtt{q}} j$ whenever $(i, j) \in Q$. This leads us to the following definitions.

Definition 4.15. Let I be a set of items, $<_{\mathtt{q}}$ any partial order on I, and $\rho \in \mathcal{R}(I)$ a ranking. We say that ρ is *consistent* with $<_{\mathtt{q}}$ if for every $i, j \in I$ with $i <_{\mathtt{q}} j$, we have $\rho(i) < \rho(j)$. ◂

Definition 4.16. Let I be a set of items. A subset R of $\mathcal{R}(I)$ is *partial-order describable (p.o.-describable)* if there is a partial order $<_{\mathtt{q}}$ on I so that R equals the set of all the rankings that are consistent with $<_{\mathtt{q}}$. A partition Λ of $\mathcal{R}(I)$ is partial-order describable if every part of Λ is p.o.-describable. ◂

In these terms, we can summarize the previous discussion.

Fact 4.17. *Let $\mathcal{T}$ be a comparison-based decision tree, and let $\Lambda_{\mathcal{T}}$ be its indistinguishability partition. Then $\Lambda_{\mathcal{T}}$ is p.o.-describable.*

As we discussed in the introduction, this leads to the partial-order certificate complexity of a function. The following definition restates Definition 4.3 from the introduction.

Definition 4.18. Let I be a set of items, and let Π be a function from $\mathcal{R}(I)$ to some codomain. The *partial-order certificate complexity (p.o.-certificate complexity)* of Π is the fewest number of parts in any partial-order describable partition Λ such that Π has a part-wise constant correct answer. ◂

We denote the measure by $\mathcal{C}_{\text{po-cert}}(\Pi)$.

Fact 4.14 implies that the p.o.-certificate complexity of a problem is a lower bound for the complexity of decisions trees that solve that problem.

Fact 4.19. *Let* Π *be any comparison-based problem. For any comparison-based decision tree* $\mathcal{T}$ *that solves* Π*, we have*

$$\mathrm{L}(\mathcal{T}) \geq \mathcal{C}_{po\text{-}cert}(\Pi) \tag{4.6}$$

$$\mathrm{ht}(\mathcal{T}) \geq \log_2\left(\mathcal{C}_{po\text{-}cert}(\Pi)\right) \tag{4.7}$$

All of our lower bounds will ultimately be lower bounds on $\mathcal{C}_{\text{po-cert}}(\Pi)$.

As we discussed in the introduction, the key players in our lower bounds are adjacent-rank transpositions (Definition 4.4). For any ranking ρ and permutation $\sigma : [n] \to [n]$ of the ranks, $\sigma \circ \rho$ is again a ranking. Adjacent-rank transpositions are the special case of this where σ is the transposition of two adjacent ranks. For concision, we will drop the '$\circ$' in $\sigma \circ \rho$ and just write $\sigma\rho$.

Before moving on, we point out one limitation of going through p.o.-certificate complexity. As the following example shows, there are problems that have low p.o.-certificate complexity, but for which every comparison-based decision tree has many leaves. In particular, both the connectivity and sensitivity approaches are subject to this limitation as they both operate by lower bounding the p.o.-certificate complexity. Nevertheless, the framework (and two approaches) turn out to be powerful enough in several settings, in particular the one of minimizing inversions in trees.

Example 4.20. Let Π be selection with rank $r = 1$ (i.e., computing the minimum). Let Λ be the partition of $\mathcal{R}(I)$ according to which item is the minimum. Each part of Λ is p.o.-describable, so Λ is as well. Since Π is (by construction) part-wise constant on Λ, we have $\mathcal{C}_{\text{po-cert}}(\Pi) \leq \#\Lambda = n$.

On the other hand, let us consider any decision tree $\mathcal{T}$ such that there exists a root-to-leaf path through $\mathcal{T}$ that makes fewer than $n - 1$ queries. Let $<_{\mathtt{q}}$ be the partial order encoding

the queries and their results along such a path. Since there were fewer than $n-1$ queries, $<_q$ does not have a unique minimal element. It follows that there are two rankings ρ_1 and ρ_2 that are both consistent with $<_q$—both lead to the same leaf of $\mathcal{T}$—yet have distinct minimums, and hence Π is not part-wise constant on $\Lambda_{\mathcal{T}}$. Consequently, in any decision tree $\mathcal{T}$ that computes Π, every root-to-leaf path must make at least $n-1$ queries. This implies that there must be at least 2^{n-1} leaves in the tree. ◂

4.4 Connectivity

In this section, we prove the Connectivity Lemma and demonstrate its utility for understanding the comparison model. As in previous sections, we fix a set I of n items. We start with a proof of the Connectivity Lemma.

Proof of the Connectivity Lemma. Let $<_q$ be the partial order such that R is the set of rankings consistent with $<_q$, and let $\rho, \rho' \in R$ be any two rankings. We need to connect ρ to ρ' by a sequence of adjacent-rank transpositions so that all the intermediate steps are in R. We do this by a recursive argument. In the base case, $\rho = \rho'$, where the empty sequence suffices.

In the recursive case, $\rho \neq \rho'$. Let $i_r \doteq \rho^{-1}(r)$ for $r = 1, \ldots, n$, and let $j_1, \ldots, j_n$ be defined similarly in terms of ρ'. As sequences, both enumerate I such that every item appears exactly once. We have depicted the sequences below to aid following the proof:

rank	1	$\cdots$	$r-1$	r	$\cdots$	t	$\cdots$	s	$\cdots$	n
ρ^{-1}	i_1	$\cdots$	i_{r-1}		$\cdots$	i_t	$\cdots$	x	$\cdots$	
ρ'^{-1}	j_1	$\cdots$	j_{r-1}	x	$\cdots$		$\cdots$		$\cdots$	

Since $\rho \neq \rho'$, the sequences differ. Let r be the least rank for which $i_r \neq j_r$; hence $i_1 = j_1$, $i_2 = j_2$, $\ldots$, and $i_{r-1} = j_{r-1}$. Let $x = j_r$, and let $s = \rho(x)$ be the rank for which $i_s = x$. Since $x \notin \{j_1, \ldots, j_{r-1}\} = \{i_1, \ldots, i_{r-1}\}$, it must be that $s \geq r$. Because the items at rank r do *not* coincide, we moreover have $s > r$. Fix any t with $r \leq t < s$. Evidently it holds that

$\rho(i_t) < \rho(x)$. Meanwhile, since $i_t \notin \{i_1 \ldots, i_{r-1}\} = \{j_1, \ldots, j_{r-1}\}$, $\rho'(i_t)$ must be at least r, and cannot equal r since $i_t \neq x$; thus $\rho'(i_t) > r = \rho'(x)$. Since ρ and ρ' are consistent with $<_{\mathtt{q}}$, we conclude that i_t and x are incomparable in $<_{\mathtt{q}}$. This holds for all t with $r \leq t < s$.

This leads us to apply the sequence of adjacent-rank transpositions $\sigma_t \doteq (t\ t+1)$ for $t = s-1, s-2, \ldots r$, in that order, to ρ, producing $\rho'' \doteq \sigma_r \sigma_{r+1} \cdots \sigma_{s-1} \rho$. Each step moves x one position to the left, and the displaced item moves one position to the right. Because x is incomparable with each of $i_r, \ldots, i_{s-1}$ in $<_{\mathtt{q}}$, consistency with $<_{\mathtt{q}}$ is preserved at every step. In the end, x is in the r-th spot, and so ρ'' agrees with ρ' on a longer prefix. Thus we recurse on ρ'' and ρ' and know that we will eventually reach the base case. ■

The following examples demonstrate the Connectivity Lemma.

Example 4.21. Let Π be the inversion parity problem. We have in context a total order $<_{\mathtt{s}}$ of the items, and we have to decide whether the number of inversions between $<_{\mathtt{s}}$ and the input ranking ρ is even or odd.

For any ρ, applying an adjacent rank transposition to ρ either increases the number of inversions by 1 or else decreases the number of inversions by 1; in either case, the parity changes. It follows that for any p.o.-describable partition on which Π is part-wise constant, there can be no two rankings ρ_1 and ρ_2 that differ by an adjacent-rank transposition yet belong to the same part of the partition. By the Connectivity Lemma, the only such partition is the trivial partition that has exactly one ranking in each part. It follows that $\mathcal{C}_{\text{po-cert}}(\Pi) \geq |\mathcal{R}(I)| = n!$. ◂

Example 4.22. Let Π be the selection problem with rank $r = \lceil n/2 \rceil$ (i.e., finding the median).

For any ranking ρ, both of the adjacent-rank transpositions that affect rank r change the median. All the other adjacent-rank transpositions do not change the median. Let G be the graph whose vertex set is $\mathcal{R}(I)$ and with edges between ρ_1 and ρ_2 whenever they differ by an adjacent rank transposition that does *not* affect rank r. The Connectivity Lemma implies that every part of a p.o.-describable partition on which Π is part-wise constant must be a

subset of a connected component in G. It follows that the p.o.-certificate complexity of Π is at least the number of connected components in G.

Meanwhile, for any two rankings ρ_1 and ρ_2, they belong to the same connected component of G if and only if they have the same median as well as the same set of items that have rank less than r (and hence also the same set of items that have rank greater than r). As there are n possibilities for the median, and, for each median, $\binom{n-1}{r-1}$ possibilities for the set of items that have rank less than r, G has $n \cdot \binom{n-1}{r-1}$ connected components. Thus we have

$$\mathcal{C}_{\text{po-cert}}(\Pi) \geq n \cdot \binom{n-1}{r-1} \tag{4.8}$$

$$= \Omega(\sqrt{n} \cdot 2^n). \tag{4.9}$$

In fact, the above characterization of the connected components of G implies that each connected component of G is p.o.-describable. Thus the partition of $\mathcal{R}(I)$ formed by taking connected components in G is p.o.-describable. Therefore (4.8) is actually an equality. ◂

Example 4.22 moreover shows how the Connectivity Lemma clarifies a subtlety in the equivalence between ordinary selection and the instantiation of partial order production that is considered equivalent to selection. Whereas selection of rank r ordinarily requires outputting only the item of rank r, the instantiation of partial order production additionally requires partitioning the remaining items according to whether their ranks are less than or greater than r. The analysis in Example 4.22, however, implies that it is impossible for the algorithm to know the item of rank r without also knowing how to partition the remaining items into those of rank less than and greater than r. It follows that, in the comparison model, ordinary selection and the instantiation of partial order production are equivalent.

4.5 Sensitivity

In this section we prove the Sensitivity Lemma. As in previous sections, we fix a set I of n items. Our proof proceeds by showing that high average sensitivity implies high p.o.-

certificate complexity.

Lemma 4.23. *Let I be a set of n items, and let $\Phi : [1, n] \to \mathbb{R}$ be any nondecreasing convex function with $\Phi(k) = k!$ for $k = 1, \ldots, n$. For any problem $\Pi : \mathcal{R}(I) \to \mathcal{O}$, the p.o.-certificate complexity of Π is related to the average sensitivity of Π by*

$$\mathcal{C}_{po\text{-}cert}(\Pi) \;\geq\; \frac{1}{n}\Phi(1 + s(\Pi)). \tag{4.10}$$

The Sensitivity Lemma follows from Lemma 4.23 by combining with Fact 4.19.

Remark 4.24. There exists a particular choice of Φ for which the Sensitivity Lemma and Lemma 4.23 are strongest, namely

$$\Phi(x) \doteq (x - \lfloor x \rfloor) \cdot (\lceil x \rceil!) + (1 - (x - \lfloor x \rfloor)) \cdot (\lfloor x \rfloor!). \tag{4.11}$$

It takes the prescribed values at integral x, and the remaining values are linearly interpolated.

◂

The remainder of this section details a proof of Lemma 4.23. We take intuition from a similar fact in the Boolean setting (cf. Exercise 8.43 from [ODo14]).

Proposition 4.25. *Let $B = \{0, 1\}^n$ be the Boolean hypercube, and let Λ be a partition of B into subcubes. For each $b \in B$, let m_b be the number of b' that differ from b by exactly one bit, yet belong to a different part of Λ than b. The dimension of the subcube containing b equals $n - m_b$. Let $\overline{m} \doteq \mathbb{E}_{b \sim B}[m_b]$. Then the number of parts of Λ must be at least $2^{\overline{m}}$.*

One way to prove this is to think of assigning to each $b \in B$ as large a weight as possible subject to ensuring that the total weight on each part of Λ is at most 1. Then the number of parts in Λ must be at least the sum of all the weights. Since the part of Λ containing b is a subcube of dimension $n - m_b$, a natural choice for the weight of b is $2^{m_b - n}$. Because the function $W(x) = 2^{x-n}$ is convex, the total weight is at least $2^n \cdot W(\overline{m})$, as desired.

Proof of Lemma 4.23. Fix Π, and fix a p.o.-describable partition Λ such that Π is part-wise constant on Λ. We need to show $\#\Lambda \geq 1/n \cdot \Phi(1 + s(\Pi))$.

For each ranking $\rho \in \mathcal{R}(I)$, let $m_\rho \in \mathbb{N}$ be the number of adjacent-rank transpositions σ such that $\sigma\rho$ and ρ are in different parts of Λ. Let $\overline{m}$ be the average value of m_ρ over a uniformly random $\rho \sim \mathcal{R}(I)$. Since Π is part-wise constant on Λ, we have

$$\begin{aligned} s(\Pi) &\le \mathbb{E}_{\rho\sim\mathcal{R}(I)}[m_\rho] \\ &= \overline{m}. \end{aligned} \tag{4.12}$$

For a fixed part R of Λ, we need to ensure that the total weight on rankings in R is not too large. The following claim, to be proven later, is the crux of this.

Claim 4.26. *Let R be any part of Λ, and let $m \in \{0, \dots, n-1\}$. The number of rankings $\rho \in R$ with $m_\rho = m$ is at most $n!/(m+1)!$.*

Let $W : [0, n-1] \to \mathbb{R}$ be the function $W(x) = \frac{1}{n!}\Phi(1+x)$. W is convex, and for integral x, $W(x) = (1+x)!/n!$. By Claim 4.26, the number of rankings ρ in R with $m_\rho = m$ is at most $1/W(m)$. The total weight of the rankings ρ with $m_\rho = m$ is thus at most 1. By summing over $m = 0, \dots, n-1$, we obtain

$$\sum_{\rho\in R} W(m_\rho) \;\le\; n. \tag{4.13}$$

Up to the factor of n, the number of parts in Λ is bounded by the sum of $W(m_\rho)$ over all $\rho \in \mathcal{R}(I)$. Thus we complete our analysis:

$$\begin{aligned} \#\Lambda &= \sum_{R\in\Lambda} 1 && (R \text{ ranges over the parts of } \Lambda) && (4.14)\\ &\ge \sum_{R\in\Lambda} \frac{1}{n} \sum_{\rho\in R} W(m_\rho) && (\text{eq. (4.13)}) && (4.15)\\ &= \frac{1}{n} \sum_{\rho\in\mathcal{R}(I)} W(m_\rho) && && (4.16)\\ &\ge \frac{1}{n} \cdot n! \cdot W(\overline{m}) && (W \text{ is convex}) && (4.17)\\ &\ge \frac{1}{n}\Phi(1+s(\Pi)). && (\text{eq. (4.12)}, \Phi \text{ is nondecreasing}) && (4.18) \end{aligned}$$

■

We now turn to proving Claim 4.26. Recall that R is the set of all rankings that are consistent with some fixed partial order $<_{\mathtt{q}}$. In terms of $<_{\mathtt{q}}$, m_ρ can be interpreted as counting the number of items i such that $\rho(i) > 1$ and such that the item j with $\rho(j) = \rho(i) - 1$ has $j <_{\mathtt{q}} i$. We say i is a *border item* in that case. We also call the item of rank 1 a border item, and no other items. Thus the number of border items is $1 + m_\rho$. Our strategy is to give a *compressed encoding* of the rankings consistent with $<_{\mathtt{q}}$ such that there is more compression as the number of border items increases. The quantitative dependence we achieved is given in the claim statement.

Our encoding is based on the well-known algorithm that, given a partial order, computes a ranking that is consistent with it. (This algorithm equivalently solves the problem of topologically sorting a directed acyclic graph.) Algorithm 2 provides pseudocode for the latter algorithm, herein named BuildRanking.

Algorithm 2

Input: A partial order $<_{\mathtt{q}}$ on a set of items I
Output: A ranking of I that is consistent with $<_{\mathtt{q}}$

```
1: procedure BuildRanking(<_q)
2:     T ← ∅
3:     S ← {i ∈ M | there is no j ∈ M ∖ T with j <_q i}
4:     for r = 1 to n do
5:         Remove an arbitrary item i from S
6:         Declare i has rank r
7:         Add i to T and update S to match
```

In our formulation, BuildRanking is nondeterministic: There is a choice to make in Line 5 for each $r = 1, \ldots, n$. As the choices vary, the resulting rankings vary as well, but always the result is consistent with $<_{\mathtt{q}}$. Through suitable choices, any ranking ρ consistent with $<_{\mathtt{q}}$ is a possible output of BuildRanking. Conversely, no two distinct sequences of choices produce the same ranking. Hence the rankings consistent with $<_{\mathtt{q}}$ are in natural bijection with the possible executions of BuildRanking.

Our encoding is a compressed description of how to make the decisions in BuildRanking such that the output is ρ. To get the encoding efficiency we seek, the critical observation

considers the behavior of BuildRanking with respect to border items. Note that, when building ρ with BuildRanking, at the beginning of the r-th iteration, T contains precisely all the items whose rank is less than r, for every r. For any border item i_0, we conclude that i_0 is not in S until just before the $r = \rho(i_0)$ step. Meanwhile, for the $r = \rho(i_0)$ step, i_0 is in S, whence BuildRanking proceeds to select $i = i_0$. In summary, as soon as a border item enters S, it is taken out and used right away. For the purposes of decoding, this means we need merely to be able to *recognize* when an item added to S is a border item; any other information concerning border items can be discarded.

This leads to the following encoding. For each ranking ρ, form the set of items that are *not* border items in ρ, and then arrange them in a sequence, L_ρ, according to the order of their ranking by ρ. Our encoding of ρ is precisely L_ρ. We can recognize the border items as those not appearing in L_ρ, while the ordering information in L_ρ tells us how to treat what remains. More precisely, given a sequence L of items, our decoder makes the decision at Line 5 of BuildRanking according to the following (mutually exclusive) rules:

1. If there are at least two items in S that do not appear in L, then abort.
2. If there is a unique item $i \in S$ that does not appear in L, then select i.
3. If every item in S appears in L, and the first item i in $L \setminus T$ is in S, then take i.
4. If every item in S appears in L, but the first item in $L \setminus T$ is *not* in S, then abort.

We show below that for every ρ, running the decoder on input L_ρ does not abort and produces ρ. Thus no two rankings consistent with $<_{\mathtt{q}}$ have the same encoding. Therefore, for each m, we can bound the number of rankings for which $m_\rho = m$ by the number of ordered sequences of $n - m - 1$ items. The number of such sequences is precisely $n!/(m+1)!$.

Proof of Claim 4.26. We have described much of the proof in the above discussion. It remains to show the correctness of the decoder.

We need to show that, for any ranking ρ consistent with $<_q$, running the decoder on L_ρ does not abort and outputs ρ. For this it suffices to show that, in each iteration, when the decoder applies its rules to make the decision on Line 5, it does not abort and its decision coincides with the decision made during the execution of BuildRanking that produces ρ.

We proceed by induction on the iteration count, r. Thus fix $r \in \{1, \ldots, n\}$, and assume all iterations prior to the r-th are successful. Let i_0 be the item with $\rho(i_0) = r$. Since the prior iterations succeeded, at the beginning of iteration r, T equals the set of items with rank less than r. In particular, all the border items with rank less than r are in T, i_0 is in S, and no other item in S is a border item. We now consider two cases.

- In the first case, i_0 is a border item. In this case, rule 2 applies, and selects $i = i_0$.
- In the second case, i_0 is not a border item. Hence, no item in S is a border item. Moreover all the items in L_ρ that precede i_0 are in T, so i_0 is the first item in $L_\rho \setminus T$. Therefore rule 3 applies and selects $i = i_0$.

In both cases, the decoder does not abort and the decoder selects $i = i_0$. This coincides with the behavior of BuildRanking while building ρ, as desired. ■

4.6 Minimizing Inversions in Trees

In this section, we apply our theory to a previously unanalyzed problem, *minimum inversions in a tree*. In this problem, we are given a rooted tree, T, as well as a ranking, ρ, of its leaves. We have the freedom to *arrange* T into an ordered tree.

Definition 4.27. For a rooted tree T, an *arrangement* of T is any way to choose, for every node v of T, a total order of the children of v. ◂

That is, an arrangement of T is any way of turning T from an unordered rooted tree into an ordered rooted tree. Each arrangement of T determines a traversal of T by visiting the root and then recursing into each of its children in their order in the arrangement. By

looking at the order in which the leaves of T are visited in the traversal, the arrangement determines a total order on the leaves; we refer to this as the order of the leaves *in the arrangement*.

With respect to the ranking ρ and an arrangement of T, an *inversion* is any pair of leaves i and j for which i appears before j in the arrangement, but $\rho(i) > \rho(j)$. In *minimum inversions in a tree*, we want to compute the minimum number of inversions that exist in some arrangement of T.

In the terms of our framework, T determines a problem Π_T. The items are the leaves of T, and thus the inputs of Π_T are the rankings ρ of the leaves of T. We overload notation and write $\mathcal{R}(T)$ to refer to the set of rankings of the leaves of T. Π_T asks us to compute, for each $\rho \in \mathcal{R}(T)$, among all the arrangements of M, what is the fewest number of inversions with respect to ρ.

Π_T is easiest when T can be arranged to put the leaves in any order. This is because it is always possible to put the leaves in the order of their rank, and thus have no inversions. As such, when T falls into this case, there is a comparison-based decision tree computing the minimum number of inversions in T with zero queries. Note that while the related problem of *finding* the minimizing arrangement in this case is identically sorting, we are only required to output the number of inversions.

Trees whose leaves may be arranged into any order play an important role in our analysis, especially as subtrees of larger trees.

Definition 4.28. A rooted tree T is *freely arrangeable* if every order of its leaves is the order in some arrangement. ◂

We can characterize the freely arrangeable rooted trees. First, we envision our trees as growing down; thus nodes are 'under', 'below', etc. their ancestors. Given a leaf of T, there is a unique path from the root to the leaf. For a nonempty set L of leaves, the associated paths have an initial segment in common; we refer to the last node in this segment as the *lowest common ancestor* (LCA) of L, and denote it by $\mathrm{LCA}(L)$. A common case is the LCA

of two (possibly indistinct) leaves ℓ_1 and ℓ_2, for which we often write $\mathrm{LCA}(\ell_1, \ell_2)$. We also define $\mathrm{LCA}(T)$ to be the LCA of all the leaves in T.

Definition 4.29. For a rooted tree T, a leaf ℓ of T, and an ancestor v of ℓ, we say that ℓ is *isolated* under v if for every other leaf ℓ' under v, we have $\mathrm{LCA}(\ell, \ell') = v$. ◂

Equivalently, ℓ is isolated under v if every ancestor of ℓ strictly between ℓ and v has exactly one child.

The freely arrangeable trees are characterized as follows.

Fact 4.30. *T is freely arrangeable if and only if all of its leaves are isolated under* $\mathrm{LCA}(T)$.

Proof. First, in the special case where T has only one leaf, there is only one order of the leaves possible, so T is necessarily fully arrangeable. Meanwhile, $\mathrm{LCA}(T)$ is the leaf, and a leaf is always isolated under itself. Thus we can assume there are $n \geq 2$ leaves, and hence $\mathrm{LCA}(T)$ is not itself a leaf.

Suppose that every leaf of T is isolated under $\mathrm{LCA}(T)$. For each leaf ℓ, let u_ℓ be the child of $\mathrm{LCA}(T)$ that is an ancestor to ℓ. Since every leaf is isolated below $\mathrm{LCA}(T)$, the u_ℓ's are all distinct. Therefore, in the traversal of T, the order in which the leaves are visited coincides with the order of the u_ℓ's in the arrangement. Since we are free to choose the latter, T is freely arrangeable.

Suppose conversely that not every leaf of T is isolated below $\mathrm{LCA}(T)$. Then there exist leaves ℓ_1, ℓ_2 such that $\mathrm{LCA}(\ell_1, \ell_2)$ lies strictly below $\mathrm{LCA}(T)$. Moreover, from the definition of $\mathrm{LCA}(T)$, there must be at least one more leaf, ℓ_3, that is below a different child of $\mathrm{LCA}(T)$ than $\mathrm{LCA}(\ell_1, \ell_2)$ is. Let u_{12} be the child of $\mathrm{LCA}(T)$ that is an ancestor to $\mathrm{LCA}(\ell_1, \ell_2)$, and let u_3 be the child of $\mathrm{LCA}(T)$ that is an ancestor to ℓ_3. In every arrangement of T, either u_{12} precedes u_3 or vice-versa. In the former case, both ℓ_1 and ℓ_2 precede ℓ_3 in the arrangement; in the latter case, ℓ_3 precedes both ℓ_1 and ℓ_2 in the arrangement. Thus no arrangement of T can put ℓ_3 between ℓ_1 and ℓ_2. We conclude that T is not freely arrangeable. ■

When T is not freely arrangeable, Π_T seems to become substantially more complex. We prove this using our lower bound framework. For some specific tree shapes, it suffices to use the Connectivity Lemma; we defer a discussion on this to Subsection 4.6.2, specifically Corollary 4.46. Our primary focus, however, will be on more general shapes, where the lower bounds follow by analyzing the sensitivity of Π_T.

As a starting point, consider that when T is freely arrangeable, the sensitivity is zero. One explanation for this is the following. Any adjacent-rank transposition involves the ranks of a pair of leaves. Because M is freely arrangeable, those leaves are both isolated under their LCA. Thus they can be freely swapped in any arrangement of T. Since swapping cancels the action of the adjacent-rank transposition, the problem is insensitive.

This reasoning carries over to general T as well. If an adjacent-rank transposition swaps the ranks of two leaves that are both isolated under their LCA, then Π_T is insensitive to it. When the input ranking is uniformly random, the affected pair of leaves are themselves uniformly random, so we are led to the following quantity.

Definition 4.31. Let T be a rooted tree. We say that two distinct leaves of T are *coisolated* if they are both isolated under their LCA.

The *coisolation probability* of T is the probability that, when selecting two distinct leaves uniformly at random from among all such pairs, they are coisolated. We denote the coisolation probability by $I^{(2)}(T)$. ◂

The preceding discussion says that when the coisolation probability of T is large, the sensitivity of Π_T must be low. The precise relationship is the following.

Fact 4.32. *For any rooted tree T, we have*

$$s\left(\Pi_T\right) \leq \left(1 - I^{(2)}\left(T\right)\right) \cdot (n-1) \tag{4.19}$$

Proof. Consider sampling a ranking ρ uniformly at random, and, independently, an adjacent-rank transposition σ uniformly at random. Let ℓ_1 and ℓ_2 be the two leaves whose ranks with

respect to ρ are changed by σ. The probability that Π_T is sensitive to σ at ρ equals $\frac{1}{n-1}s(\Pi_T)$. Meanwhile, as discussed above, if ℓ_1 and ℓ_2 are both isolated under their LCA, then Π_T is insensitive to σ. It follows that $\frac{1}{n-1}s(\Pi_T)$ and $I^{(2)}(T)$ measure the probabilities of disjoint events, and so their sum must be at most 1. The fact now follows by rearranging. ■

Whereas Fact 4.32 says $I^{(2)}(T)$ determines an upper bound on the sensitivity, we show that, at least in a special case, the upper bound is also not far from a lower bound. Our proof requires us to restrict to trees in which every subtree that is not freely arrangeable has top fan-in at most two. Freely arrangeable subtrees are still permitted arbitrary fan-in.

Theorem 4.33. *Let T be a rooted tree with n leaves such that the root of every subtree that is not freely arrangeable has at most two children. Then the average sensitivity of the minimum number of inversions among arrangements of T satisfies*

$$\nu(T) \;\geq\; s(\Pi_T) \;\geq\; \frac{1}{3}\nu(T), \tag{4.20}$$

where $\nu(T) \doteq (1 - I^{(2)}(T)) \cdot (n-1)$.

Combined with the Sensitivity Lemma we obtain the following lower bound.

Theorem 4.34. *Let T be a rooted tree with n leaves such that the root of every subtree that is not freely arrangeable has at most two children, and let $\Phi : [1, n] \to \mathbb{R}$ be any nondecreasing convex function with $\Phi(k) = k!$ for $k = 1, \ldots, n$. Every comparison decision tree $\mathcal{T}_{\text{dec}}$ that computes the minimum number of inversions among arrangements of T satisfies*

$$\mathrm{L}(\mathcal{T}_{\text{dec}}) \;\geq\; \frac{1}{n}\Phi\left(1 + \frac{1 - I^{(2)}(T)}{3}(n-1)\right), \tag{4.21}$$

and hence

$$\mathrm{ht}(\mathcal{T}_{\text{dec}}) \;\geq\; \log_2 \Phi\left(1 + \frac{1 - I^{(2)}(T)}{3}(n-1)\right) - \log_2(n). \tag{4.22}$$

Before moving on to a proof of Theorem 4.34, we apply the theorem to the special case where T is a binary tree.

Corollary 4.35. *Let T be a binary tree. Every comparison decision tree $\mathcal{T}_{\text{dec}}$ that computes the minimum number of inversions among arrangements of T satisfies*

$$\mathrm{L}(\mathcal{T}_{\text{dec}}) \geq \frac{1}{n}\left\lceil \frac{n}{3} \right\rceil !, \tag{4.23}$$

and hence

$$\mathrm{ht}(\mathcal{T}_{\text{dec}}) \geq \log_2 \left\lceil \frac{n}{3} \right\rceil ! - \log_2 n \tag{4.24}$$

$$= \Omega(n \log n). \tag{4.25}$$

Proof. Since T is binary, for any leaf ℓ, there is at most one other leaf ℓ' such that ℓ and ℓ' are coisolated. It follows that the coisolation probability for binary trees is at most $1/(n-1)$. Corollary 4.35 now follows from Theorem 4.34. ∎

Thus the number of queries required to compute the minimum number of inversions in binary trees is, up to constants, on par with sorting. Later, we will see Corollaries 4.46 and 4.49, which provide stronger bounds in special cases.

4.6.1 Proof of Theorem 4.33

The remainder of this section is dedicated to proving Theorem 4.33. We begin by observing that Π_T can be decomposed into simpler subproblems. Recall that an arrangement of T consists of a choice for each node v in T, how to order of the children of v. Importantly, the choices are independent across different v. Meanwhile, for any pair of distinct leaves ℓ_1, ℓ_2, whether the pair constitutes an inversion depends on exactly one of the choices, namely the choice for $v = \mathrm{LCA}(\ell_1, \ell_2)$. It follows that minimizing the overall number of inversions is equivalent to minimizing *separately*, for each v in T, the number of inversions whose LCA is v.

Thus we define the following quantities, one for each v in T, that capture the subproblem at v.

Definition 4.36. Let T be a rooted tree, and let v be a node in T. Relative to a ranking ρ of the leaves of T and an arrangement of T, a *cross-inversion* at v is any pair of distinct leaves ℓ_1, ℓ_2 with $\mathrm{LCA}(\ell_1, \ell_2) = v$ such that ℓ_1 precedes ℓ_2 in the arrangement and $\rho(\ell_1) > \rho(\ell_2)$.

Define $X_{T,v}$ to be the function that takes in a ranking ρ of the leaves of T and outputs the minimum, over arrangements of T, of the number of cross-inversions at v with respect to ρ. ◂

To emphasize, when selecting an arrangement of T, the only choice that can influence the number of cross-inversions at v is how the arrangement orders the children of v.

The following fact formalizes the observations above.

Fact 4.37. *Let T be a rooted tree. For every ranking ρ of the leaves of T, we have*

$$\Pi_T(\rho) = \sum_{v \in T} X_{T,v}(\rho). \tag{4.26}$$

Whereas this decomposition is manifestly useful for algorithm design, it is also useful for studying sensitivity. This is because the application of any one adjacent-rank transposition can change at most one term in (4.26). Namely, if the adjacent-rank transposition changes the ranks of leaves ℓ_1 and ℓ_2, then only the term for $v = \mathrm{LCA}(\ell_1, \ell_2)$ can change.

This suggests we analyze the sensitivity on a "node-by-node basis". To make sense of this, we first recall the distribution used in the proof of Fact 4.32, as it will be useful throughout the proof of Theorem 4.33.

Definition 4.38. For a rooted tree T, let $\mathcal{D}_T$ denote the uniform distribution on pairs (ρ, σ) where ρ is a ranking of the leaves of T and σ is an adjacent-rank transposition.

We moreover define the related random variables ℓ_{lo} and ℓ_{hi} to be the leaves of T with $\rho(\ell_{\mathrm{lo}}) < \rho(\ell_{\mathrm{hi}})$ such that $\sigma = (\rho(\ell_{\mathrm{lo}})\ \rho(\ell_{\mathrm{hi}}))$. We refer to these as the *affected leaves* of σ with respect to ρ. ◂

As we stated in the proof of Fact 4.32, $\frac{1}{n-1} s(\Pi_T)$ equals the probability that Π_T is sensitive to σ at ρ when ρ and σ are drawn from $\mathcal{D}_T$. This probability can be broken up according to

which node is $\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}})$. Together with Fact 4.37, we have the following.

$$\frac{1}{n-1} s(\Pi_T) = \sum_{v \in T} \Pr\left[\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v \text{ and } \Pi_T(\rho) \neq \Pi_T(\sigma\rho)\right] \tag{4.27}$$

$$= \sum_{v \in T} \Pr\left[\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v \text{ and } X_{T,v}(\rho) \neq X_{T,v}(\sigma\rho)\right] \tag{4.28}$$

For v that have fewer than two children, the probability constituting the v-th term above is necessarily zero, because v can never be the LCA of two distinct leaves. Likewise, if the subtree rooted at v is freely arrangeable, then the probability is zero, because $X_{T,v}(\rho)$ is zero for every ρ. For v that do not fall into these two cases, we can decompose the v-th term using the following quantities.

Definition 4.39. Let T be a rooted tree with n leaves, and let v be a node in T such that v has at least two children and such that the subtree of T rooted at v is not freely arrangeable. We name the following probabilities, where ρ, σ are drawn from $\mathcal{D}_T$, and ℓ_{lo} and ℓ_{hi} are the affected leaves.

$$p_{T,v} \doteq \Pr[\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v] \tag{4.29}$$

$$I^{(2)}_{T,v} \doteq \Pr[\ell_{\mathrm{lo}} \text{ and } \ell_{\mathrm{hi}} \text{ are coisolated} \mid \mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v] \tag{4.30}$$

$$\hat{s}_{T,v} \doteq \Pr[X_{T,v}(\rho) \neq X_{T,v}(\sigma\rho) \mid \mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v \;\&\; \ell_{\mathrm{lo}}, \ell_{\mathrm{hi}} \text{ are not coisolated}] \tag{4.31}$$

◂

The following lemma collects the preceding discussion.

Lemma 4.40. *Let T be a rooted tree with n leaves. With respect to the quantities defined in Definition 4.39, we have*

$$s(\Pi_T) = (n-1) \sum_v p_{T,v} (1 - I^{(2)}_{T,v}) \hat{s}_{T,v} \tag{4.32}$$

where in the sum v ranges over the nodes in T such that v has at least two children and such that the subtree rooted at v is not freely arrangeable.

We note also that the coisolation probability of T equals the probability that ℓ_{lo} and ℓ_{hi} are coisolated, which is equivalently the sum of $p_{T,v} \cdot I^{(2)}_{T,v}$ over the v in T where those quantities are defined. In symbols, we have

$$I^{(2)}(T) = \sum_v p_{T,v} I^{(2)}_{T,v}, \tag{4.33}$$

where again v ranges over the nodes of T such that the subtree rooted at v is not freely arrangeable and such that v has at least two children.

From here, we analyze the quantities in Lemma 4.40. The trickiest quantities are the $\hat{s}_{T,v}$. The following lemma establishes the bound we obtain in this section.

Lemma 4.41. *Let T be a rooted tree, and let v be a node in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable. Then*

$$\hat{s}_{T,v} \geq \frac{1}{3} \tag{4.34}$$

We prove Lemma 4.41 in the next subsection. In the meantime, it is a short calculation to prove Theorem 4.33 from Lemma 4.41.

Proof of Theorem 4.33. Lemma 4.41 says each of the quantities $\hat{s}_{T,v}$ in Definition 4.39 is bounded below by 1/3. By Lemma 4.40, we have

$$s(\Pi_T) \geq \frac{1}{3}(n-1) \sum_v p_{T,v} \cdot (1 - I^{(2)}_{T,v}) \tag{4.35}$$

$$= \frac{1}{3}(n-1)(1 - I^{(2)}(T)), \tag{4.36}$$

which combined with Fact 4.32 proves Theorem 4.33. ∎

4.6.2 Proof of Lemma 4.41

It now remains to prove Lemma 4.41. To analyze the quantities $\hat{s}_{T,v}$, our first step is to characterize the event that underlies $\hat{s}_{T,v}$. We do this in Lemma 4.44 below. The characterization applies to the particular case where v has exactly two children.

In order to state Lemma 4.44, it will be helpful to have the following notation for counting cross inversions.

Definition 4.42. Let T be a rooted tree, and let ℓ_1 and ℓ_2 be two distinct leaves of T. For a ranking ρ of the leaves of T, we define

$$\delta_\rho(\ell_1 : \ell_2) \doteq \begin{cases} +1 & \text{if } \rho(\ell_1) < \rho(\ell_2) \\ -1 & \text{if } \rho(\ell_1) > \rho(\ell_2) \end{cases}. \tag{4.37}$$

For two disjoint sets A, B of leaves, set

$$\delta_\rho(A : B) \doteq \sum_{\substack{\ell_1 \in A \\ \ell_2 \in B}} \delta_\rho(\ell_1 : \ell_2). \tag{4.38}$$

◂

For any arrangement of T, when ℓ_1 precedes ℓ_2 in the arrangement, $\delta_\rho(\ell_1 : \ell_2)$ takes the value -1 when those leaves constitute an inversion, and takes the value $+1$ otherwise. Note that $\delta_\rho(\ell_1 : \ell_2) = -\delta_\rho(\ell_2 : \ell_1)$ as well as $\delta_\rho(A : B) = -\delta_\rho(B : A)$. It is also true that $\delta_\rho(A : B) \equiv |A||B| \pmod 2$.

It will also be helpful to name a few other quantities associated to samples from $\mathcal{D}_T$.

Definition 4.43. Let T be a rooted tree. Let ρ be a ranking of the leaves of T, let σ be an adjacent-rank transposition, let ℓ_{lo} and ℓ_{hi} be the affected leaves. Let u_{lo} be the child of $\text{LCA}(\ell_{\text{lo}}, \ell_{\text{hi}})$ that is an ancestor of ℓ_{lo}, and let u_{hi} be the same with respect to ℓ_{hi}, where note $u_{\text{lo}} \neq u_{\text{hi}}$. We name the following sets:

- A is the set of leaves below u_{lo} except ℓ_{lo}.
- B is the set of leaves below u_{hi} except ℓ_{hi}.
- $A_<$ is the set of leaves in A that have rank less than $\rho(\ell_{\text{lo}})$.
- $A_>$ is the set of leaves in A that have rank greater than $\rho(\ell_{\text{lo}})$.

- $B_<$ is the set of leaves in B that have rank less than $\rho(\ell_{\text{hi}})$.
- $B_>$ is the set of leaves in B that have rank greater than $\rho(\ell_{\text{hi}})$.

When ρ and σ are sampled from $\mathcal{D}_T$, the above sets are likewise random variables. ◂

Note that since ℓ_{lo} and ℓ_{hi} have adjacent ranks, every leaf in A is in exactly one of $A_<$ or $A_>$, and every leaf in B is in exactly one of $B_<$ or $B_>$.

Lemma 4.44. *Let T be a rooted tree, and let v be a node in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable. For any ranking ρ of the leaves of T and any adjacent-rank transposition σ, we have $X_{T,v}(\rho) \neq X_{T,v}(\sigma\rho)$ if and only if*

$$\delta_\rho(A:B) \neq -|A_<| + |A_>| + |B_<| - |B_>|, \tag{4.39}$$

where the quantities in (4.39) are those defined in Definition 4.43.

Proof. Since v has exactly two children, there are only two ways to order them. $X_{T,v}$ is the fewer number cross inversions among the two options. In terms of the notation in Definition 4.42, we thus have

$$X_{T,v}(\rho') = \min\left(\frac{|A^+||B^+| - \delta_{\rho'}(A^+:B^+)}{2}, \frac{|B^+||A^+| - \delta_{\rho'}(B^+:A^+)}{2}\right), \tag{4.40}$$

where $A^+ \doteq A \cup \{\ell_{\text{lo}}\}$, and $B^+ \doteq B \cup \{\ell_{\text{hi}}\}$. Writing the minimum as the average of its two inputs minus half their difference, we obtain

$$X_{T,v}(\rho') = \frac{1}{2}|A^+||B^+| - \frac{1}{4}\left|\delta_{\rho'}(A^+:B^+) - \delta_{\rho'}(B^+:A^+)\right| \tag{4.41}$$

$$= \frac{1}{2}|A^+||B^+| - \frac{1}{2}\left|\delta_{\rho'}(A^+:B^+)\right|, \tag{4.42}$$

where we have used that $\delta_{\rho'}(B^+:A^+) = -\delta_{\rho'}(A^+:B^+)$. Restricting ρ' to one of the values in $\{\rho, \sigma\rho\}$, we compute the second term as follows.

$$\delta_{\rho'}(A^+:B^+) = \delta_{\rho'}(A:B) + \delta_{\rho'}(\ell_{\text{lo}}:B) + \delta_{\rho'}(A:\ell_{\text{hi}}) + \delta_{\rho'}(\ell_{\text{lo}}:\ell_{\text{hi}}) \tag{4.43}$$

$$= V - [|B_<| - |B_>| - |A_<| + |A_>|] + \delta_{\rho'}(\ell_{\text{lo}}:\ell_{\text{hi}}) \tag{4.44}$$

where $V \doteq \delta_\rho(A:B) = \delta_{\sigma\rho}(A:B)$ is independent of the choice of ρ'. It follows that the dependence on ρ' in $X_{T,v}(\rho')$ is limited to the term $\delta_{\rho'}(\ell_{\text{lo}}:\ell_{\text{hi}})$ in an absolute value. Since we have $\delta_\rho(\ell_{\text{lo}}:\ell_{\text{hi}}) = -\delta_{\sigma\rho}(\ell_{\text{lo}}:\ell_{\text{hi}}) \neq 0$, it follows that $X_{T,v}(\rho) = X_{T,v}(\sigma\rho)$ if and only if the other terms in the absolute value are zero. Up to rearranging, this is precisely (4.39). ∎

Our proof of Lemma 4.41 relies on independence among the quantities in (4.39) when ρ and σ are drawn from $\mathcal{D}_T$. It is easiest to see the independence when the samples are generated in a particular way, namely according to SAMPLE–$\mathcal{D}_T$ as described in Algorithm 3. The following fact asserts the correctness of the sampler.

Fact 4.45. *For every ordered tree T,* SAMPLE–$\mathcal{D}_T$ *samples $\mathcal{D}_T$.*

Proof. Every possible execution of SAMPLE–$\mathcal{D}_T$ produces a valid ranking of the leaves of T, a valid adjacent-rank transposition, and the correct pair of affected leaves. Every ranking and adjacent-rank transposition (and thus also the pair of affected leaves) is output by exactly one possible execution of SAMPLE–$\mathcal{D}_T$. Since every possible execution is equally likely, it follows that SAMPLE–$\mathcal{D}_T$ samples the uniform distribution over all pairs of rankings and adjacent-rank transpositions, and this is precisely $\mathcal{D}_T$. ∎

Algorithm 3

Input: An ordered tree T with at least two leaves.

Output: A ranking ρ of the leaves of T, an adjacent-rank transposition σ, and the affected leaves ℓ_{lo} and ℓ_{hi}, all sampled according to $\mathcal{D}_T$.

procedure SAMPLE–$\mathcal{D}_T$
  Set $\ell_{\text{lo}}, \ell_{\text{hi}} \leftarrow$ uniformly random pair of distinct leaves in T
  Set $v \leftarrow \text{LCA}(\ell_{\text{lo}}, \ell_{\text{hi}})$
  Set $R \leftarrow$ a list containing every leaf of T that is below v, except for ℓ_{lo} and ℓ_{hi}, in a uniformly random order
  Insert the symbol $*$ at a uniformly random location in R
  For each leaf of T not below v, insert it into R at a uniformly random location
  Replace the $*$ in R by ℓ_{lo} and ℓ_{hi}, with ℓ_{lo} before ℓ_{hi}
  Set $\rho \leftarrow$ the ranking where the r-th leaf in R has rank r
  Set $\sigma \leftarrow (\rho(\ell_{\text{lo}})\ \rho(\ell_{\text{hi}}))$
  Output ρ, σ, ℓ_{lo}, and ℓ_{hi}

To prove Lemma 4.41, we focus on the random choice made in Line 5 of SAMPLE–$\mathcal{D}_T$. Following the completion of Line 4, the left-hand side of (4.39) is fixed, and the right-hand side will be determined subsequent to Line 5. By considering (4.39) modulo 2 and 4, we can argue that $\hat{s}_{T,v}$ is either necessarily 1, or else must be at least a quantity slightly less than half when v has many nodes below it. Accounting for cases where v has few leaves below it, we get the bound of 1/3.

Proof of Lemma 4.41. Fix v as in the lemma statement. We consider the execution of SAMPLE–$\mathcal{D}_T$ when conditioning on ℓ_{lo} and ℓ_{hi} subject to $\text{LCA}(\ell_{\text{lo}}, \ell_{\text{hi}}) = v$, and conditioning on the order for R after Line 4. The left-hand side of (4.39) is determined, while the right-hand side depends on the choice in Line 5. Critically, as the position of the $*$ varies by one spot, either $|A_<| - |A_>|$ or $|B_<| - |B_>|$ changes by exactly 2 (up or down) and the other does not change. Thus, modulo 4, the right-hand side of (4.39) changes by exactly 2. It follows that the left- and right-hand sides of (4.39) must be different from any particular value for at least half of the possible executions at Line 5, rounded down. Thus we have the bound

$$\hat{s}_{T,v} \geq \frac{1}{2} - \frac{1}{2(|A| + |B| + 1)}. \tag{4.45}$$

Since v is not freely arrangeable, we have $|A| + |B| \geq 1$. By considering (4.39) modulo 2, the left-hand side is $|A||B|$, and the right-hand side is $|A| + |B|$. If $|A| + |B| = 1$, we conclude $\hat{s}_{T,v} = 1$. Thus we can take $|A| + |B| \geq 2$ in (4.45), leading to $\hat{s}_{T,v} \geq 1/3$ as desired. ∎

In the last step of the preceding proof, we see that there are circumstances where $\hat{s}_{T,v}$ must actually equal 1, in particular whenever either $|A|$ or $|B|$ is odd. For trees where these circumstances always apply, we can actually prove a strong lower bound based on the Connectivity Lemma, with no use of our sensitivity machinery. We demonstrate this in the case of perfect binary trees.

Corollary 4.46. *Let T be a perfect binary tree with n leaves. Every comparison-based decision tree $\mathcal{T}_{\text{dec}}$ that computes the minimum number of inversions among arrangements*

of T satisfies

$$\mathrm{L}(\mathcal{T}_{\mathrm{dec}}) \geq \frac{n!}{2^{n/2}}, \tag{4.46}$$

and hence

$$\mathrm{ht}(\mathcal{T}_{\mathrm{dec}}) \geq \log_2(n!) - n/2. \tag{4.47}$$

Compared to Corollary 4.35, the bound in Corollary 4.46 is stronger, close to the bound obtained by Corollary 4.49, and the argument is simpler than both of those. However, unlike Corollaries 4.35 and 4.49, the argument is fragile and does not generalize to as many tree shapes.

Proof. Let Λ be an arbitrary partial-order describable partition of the rankings of the leaves of T on which the minimum number of inversions, Π_T, is part-wise constant. We show that no part of Λ can have more than $2^{n/2}$ rankings, and therefore $\mathcal{C}_{\text{po-cert}}(\Pi_T)$ must be at least $n!/2^{n/2}$. Corollary 4.46 then follows by Fact 4.19.

Let ρ be any ranking of the leaves of T and let σ be an adjacent-rank transposition. Let ℓ_{lo} and ℓ_{hi} be the affected leaves, and let v be their LCA. If v is in the layer of T just above the leaves (i.e., ℓ_{lo} and ℓ_{hi} are siblings), then the subtree rooted at v is freely arrangeable, and the minimum number of inversions is the same with respect to both ρ and $\sigma\rho$. Therefore we consider the case where v is in a higher layer of T.

In this case, because T is a perfect binary tree, both children of v have an even number of leaves beneath them. In terms of Lemma 4.44, the sets A and B have odd cardinality, since they respectively exclude ℓ_{lo} and ℓ_{hi}. Considering (4.39) modulo 2, it follows that $X_{T,v}(\rho) \neq X_{T,v}(\sigma\rho)$, and hence $\Pi_T(\rho) \neq \Pi_T(\sigma\rho)$. We conclude that ρ and $\sigma\rho$ must belong to different parts of Λ.

In other words, the only adjacent-rank transpositions that connect rankings in the same part of Λ are those that swap two leaves that are siblings. From any ranking, there are at most $2^{n/2}$ rankings that can be reached by such adjacent-rank transpositions. By the Connectivity Lemma, each part of Λ has at most $2^{n/2}$ rankings in it, as desired. ∎

4.7 Refined Sensitivity Analysis

In this section, we present a refinement to the sensitivity analysis in Section 4.6. The result is an improvement from the multiplicative loss of 1/3 in Theorem 4.33 to an additive loss of only $O(1)$ under the hypothesis of a combinatorial conjecture, Conjecture 4.58.

Theorem 4.47. *Let T be a rooted tree with n leaves such that the root of every subtree that is not freely arrangeable has at most two children. Assuming Conjecture 4.58, the following relationship holds:*

$$\nu(M) \;\geq\; s(\Pi_M) \;\geq\; \nu(M) - O(1) \tag{4.48}$$

where $\nu(M) = (1 - I^{(2)}(T)) \cdot (n-1)$.

Conjecture 4.58 arises in the following way. In the proof of Theorem 4.47, we are led to consider the distribution of $\delta_\rho(A : B)$ where A and B are fixed sets of leaves, and ρ is drawn uniformly at random from rankings of $A \sqcup B$. For Theorem 4.47, we need an upper bound on the probability that $\delta_\rho(A : B) = k$ that is independent of k and that decreases as A or B grows. The distribution is closely related to the Gaussian binomial coefficients, which are important—but notoriously difficult—quantities in combinatorics. A general bound suitable for our purposes does not seem to be known. Heuristic arguments as well as rigorous results in special cases consistently suggest a particular bound ought to hold, and Conjecture 4.58 asserts that this bound holds in general. We discuss Conjecture 4.58 in more detail in Subsection 4.7.3.

Because Conjecture 4.58 is known in some special cases, we can show unconditional sensitivity lower bounds for rooted trees that satisfy additional hypotheses.

Theorem 4.48. *For any constants $\alpha > 0, L \geq 1$, there is a constant c so that the following holds. Let T be a rooted tree with n leaves such that, for every node v in T, either the subtree rooted at v is freely arrangeable, or else v has at most two children and either*

1. *each child of v is an ancestor of at least an α fraction of the leaves below v, or*

2. at least one child of v has at most L leaves below it.

Then the following relationships hold:

$$\nu(T) \;\geq\; s(\Pi_T) \;\geq\; \nu(T) - c, \tag{4.49}$$

where $\nu(M) = (1 - I^{(2)}(T)) \cdot (n-1)$.

Under the hypotheses of either theorem, lower bounds on the complexity of decision trees that compute the minimum inversions in a rooted tree follow by composing with the Sensitivity Lemma.

The following corollary demonstrates the strength of the bounds obtained.

Corollary 4.49. *Let T be a perfect binary tree. Then any decision tree $\mathcal{T}_{\text{dec}}$ that computes the minimum number of inversions among arrangements of T satisfies*

$$\begin{aligned} \mathrm{L}(\mathcal{T}_{\text{dec}}) &\geq \tfrac{1}{n} \cdot \lfloor n - O(1) \rfloor! \\ &= n! \cdot \tfrac{1}{n^{O(1)}} \end{aligned} \tag{4.50}$$

$$\mathrm{ht}(\mathcal{T}_{\text{dec}}) \geq \log_2 n! - O(\log n).$$

Proof. The hypotheses of Theorem 4.48 apply with $\alpha = 1/2$, and, as we computed in the proof of Corollary 4.35, the coisolation probability is $1/(n-1)$. Corollary 4.49 now follows by composing Theorem 4.48 with the Sensitivity Lemma. ∎

The lower bound on the decision tree height is only a small additive amount away from the lower bound for sorting, a significant improvement over the constant multiplicative loss of Corollary 4.35. The argument is also extends (assuming Conjecture 4.58 if necessary) to more general tree shapes than Corollary 4.46.

4.7.1 Proofs of Theorems 4.47 and 4.48

We now turn to proving Theorems 4.47 and 4.48. As in the proof of Theorem 4.33, we use Lemma 4.40 to decompose the sensitivity into the simpler quantities $p_{T,v}$, $I^{(2)}_{T,v}$, and $\hat{s}_{T,v}$, and

then use the characterization in Lemma 4.44 to derive a lower bound for the quantities $\hat{s}_{T,v}$. The proofs diverge from Theorem 4.33 in the precise lower bound on $\hat{s}_{T,v}$. The following lemma gives the bound used for Theorems 4.47 and 4.48.

Lemma 4.50. *Let T be a rooted tree, and let v be a node in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable. Let $n_{v,1}$ and $n_{v,2}$ be the number of leaves below each child of v. Assuming that either Conjecture 4.58 holds, or else Conditions 1 and 2 from Theorem 4.48 hold for v, there is a constant c such that*

$$\hat{s}_{T,v} \geq 1 - \frac{c}{\sqrt{n_{v,1} n_{v_2} (n_{v,1} + n_{v,2})}}. \tag{4.51}$$

We will prove Lemma 4.50 in Subsection 4.7.2. For now, we show how to complete the proofs of Theorems 4.47 and 4.48. At a high level, we proceed similarly to Theorem 4.33, but some more sophistication is necessary. Whereas in Theorem 4.33 we bounded each $\hat{s}_{T,v}$ by the worst-case over all v, doing so here would miss out on the fact that the lower bound from Lemma 4.50 is close to 1 when v has many leaves. Since the quantities $p_{T,v}$ weight the v with more leaves higher than those with fewer leaves, we can expect to capitalize on this asymptotic behavior.

Proof of Theorem 4.47. From Lemma 4.40 we have

$$s(\Pi_T) = (n-1) \sum_v p_{T,v} (1 - I^{(2)}_{T,v}) \hat{s}_{T,v} \tag{4.52}$$

$$= (n-1)(1 - I^{(2)}(T)) - (n-1) \sum_v p_{T,v} (1 - I^{(2)}_{T,v})(1 - \hat{s}_{T,v}) \tag{4.53}$$

where the sums range over all the nodes v in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable.

For each such v, let $n_{v,1}$ and $n_{v,2}$ be the number of leaves below each child of v. We

bound the second term in (4.53) according to the following relationships.

$$p_{T,v} = \frac{n_{v,1}n_{v,2}}{\binom{n}{2}} \tag{4.54}$$

$$I^{(2)}_{T,v} \geq 0 \tag{4.55}$$

$$\hat{s}_{T,v} \geq 1 - \frac{c}{\sqrt{n_{v,1}n_{v,2}(n_{v,1}+n_{v,2})}} \tag{4.56}$$

(4.54) counts how many pairs of leaves have v as their LCA over the total number of pairs of leaves. (4.55) is trivial. (4.56) is Lemma 4.50.

Plugging these into (4.53), we have

$$s(\Pi_T) \geq (n-1)(1-I^{(2)}(T)) - \frac{2c}{n}\sum_v \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}}. \tag{4.57}$$

The following bound then completes the theorem.

Claim 4.51. *There is a constant c', independent of T, such that*

$$\sum_v \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}} \leq c'n. \tag{4.58}$$

Plugged into (4.57), we have

$$s(\Pi_T) \geq (n-1)(1-I^{(2)}(T)) - 2cc' \tag{4.59}$$

as desired. ∎

It remains to prove Claim 4.51.

Proof of Claim 4.51. For each node w of T, let T_w be the subtree rooted at w, and let n_w be the number of leaves below w. We show that the following holds for all w in T

$$\sum_{v\in T_w} \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}} \leq \left(\frac{1+\sqrt{2}}{2}\right)n_w - \left(\frac{1+\sqrt{2}}{2}\right)\sqrt{n_w} \tag{4.60}$$

where the sum ranges over $v \in T_w$ that have exactly two children and such that the subtree rooted at v is not freely arrangeable. Claim 4.51 follows from the case where w is the root of T.

We show (4.60) by structural induction on T_w. The base case is when T_w is freely arrangeable, such as when w is a leaf. In this case, the left-hand side of (4.60) is zero and the right-hand side is non-negative, so (4.60) holds. For the inductive step, T_w is not freely arrangeable. We branch into two cases. In the first case, w has only one child, w'. In this case, the left- and right-hand sides of (4.60) are the same for w as w'. (4.60) for w therefore follows from the inductive hypothesis on w'. In the second case, w has exactly two children. There is thus a term where $v = w$ on the left-hand side of (4.60). The remaining terms in that sum can be grouped according to which child of w is an ancestor of v. These two groups have the form (4.60) where the root is the relevant child of w. Applying the inductive hypotheses to those groups, we have the following.

$$\begin{aligned}\sum_{v \in T_w} \sqrt{\frac{n_{v,1} n_{v,2}}{n_{v,1} + n_{v,2}}} &\leq \sqrt{\frac{n_{w,1} n_{w,2}}{n_{w,1} + n_{w,2}}} \\ &\quad + \left(\frac{1+\sqrt{2}}{2}\right) n_{w,1} - \left(\frac{1+\sqrt{2}}{2}\right) \sqrt{n_{w,1}} \\ &\quad + \left(\frac{1+\sqrt{2}}{2}\right) n_{w,2} - \left(\frac{1+\sqrt{2}}{2}\right) \sqrt{n_{w,2}} \qquad (4.61) \\ &= \left(\frac{1+\sqrt{2}}{2}\right) n_w - \left(\frac{1+\sqrt{2}}{2}\right) \sqrt{n_w} \cdot F\left(\frac{n_{w,1}}{n_w}\right) \qquad (4.62)\end{aligned}$$

where

$$F(x) \doteq \sqrt{x} + \sqrt{1-x} - (2\sqrt{2} - 2)\sqrt{x(1-x)}. \qquad (4.63)$$

From here, it suffices for $F(x)$ to be at least 1 for all x with $0 \leq x \leq 1$. Because F is continuous, it attains a minimum on $[0, 1]$. On $(0, 1)$, F is differentiable, and its derivative vanishes only at

$$x \in \left\{ \frac{1}{8}\left(4 - \sqrt{7 + 4\sqrt{2}}\right), \frac{1}{2}, \frac{1}{8}\left(4 + \sqrt{7 + 4\sqrt{2}}\right) \right\}. \qquad (4.64)$$

Thus its minimum is at one of those points or at $x = 0$ or $x = 1$. For $x \in \{0, 1/2, 1\}$, $F(x) = 1$, while for the other two x, $F(x) \approx 1.0177 > 1$. ∎

4.7.2 Proof of Lemma 4.50

In this subsection we prove Lemma 4.50. Actually, we prove a variant, Lemma 4.54, that does not assume Conjecture 4.58 nor Conditions 1 and 2 from Theorem 4.48, and instead concludes with a more technical bound on $\hat{s}_{T,v}$. Conjecture 4.58 translates the latter into the form asserted by Lemma 4.50. Conditions 1 and 2 from Theorem 4.48 capture conditions under which Conjecture 4.58 is known to hold, and hence the same translation goes through.

In order to state Lemma 4.54, we need the following definitions.

Definition 4.52. For any two positive integers $a, b \geq 0$ and any integer k, define

$$g(a,b,k) \doteq \Pr\left[\delta_\rho\left(A : B\right) = k\right] \tag{4.65}$$

where A and B are an arbitrary pair of disjoint sets of items with $|A| = a$ and $|B| = b$, and ρ is a uniformly random ranking $A \sqcup B \to [a+b]$. ◂

Definition 4.53. For any real number $\eta \in (0,1)$ and positive integers $\hat{a}, \hat{b} \geq 0$, define

$$\gamma_\eta\left(\hat{a},\hat{b}\right) \doteq \max_{a,b} \max_{k} g(a,b,k) \tag{4.66}$$

where a and b range over integers satisfying $\eta\hat{a} \leq a \leq \hat{a}$ and $\eta\hat{b} \leq b \leq \hat{b}$, and k ranges over the integers. ◂

Lemma 4.54. *Let T be a rooted tree, and let v be a node in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable. Let $n_{v,1}$ and $n_{v,2}$ be the number of leaves below each child of v.*

- *If $\min(n_{v,1}, n_{v,2}) = 1$, then $\hat{s}_{T,v} \geq 1 - \frac{1}{m}$ where $m = \max(n_{v,1}, n_{v,2})$.*
- *If $\min(n_{v,1}, n_{v,2}) \geq 2$, then $\hat{s}_{T,v} \geq 1 - \gamma_{1/4}\left(n_{v,1} - 1, n_{v,2} - 1\right)$.*

We treat the $\min(n_{v,1}, n_{v,2}) = 1$ case separately from the rest because the bound in the $\min(n_{v,1}, n_{v,2}) \geq 2$ case degenerates if either of $n_{v,1} - 1$ or $n_{v,2} - 1$ becomes zero. In particular,

if $\min(\hat{a}, \hat{b})$ is zero, then $1 - \gamma_{1/4}\left(\hat{a}, \hat{b}\right)$ is 0, independent of $\max(\hat{a}, \hat{b})$. We would rather have a lower bound on $\hat{s}_{T,v}$ that goes to 1 as $\max(n_{v,1}, n_{v,2})$ grows.

When $\min(n_{v,1}, n_{v,2}) = 1$, Lemma 4.50 follows directly from Lemma 4.54. Otherwise, as explained above, we bound $\gamma_{1/4}(n_{v,1} - 1, n_{v,2} - 1)$ using either Conjecture 4.58 or its proven special cases. The bounds are formalized in Fact 4.59 and Theorem 4.60 in Subsection 4.7.3.

The remainder of this subsection is dedicated to proving Lemma 4.54. As in the proof of Lemma 4.41, we base our analysis on the characterization in Lemma 4.44. In the case where $\min(n_{v,1}, n_{v,2}) = 1$, we proceed along similar lines as the proof of Lemma 4.41. Here is the proof when $n_{v,1} = 1$; the case where $n_{v,2} = 1$ is symmetric.

Proof of Lemma 4.54, when $n_{v,1} = 1$. Consider sampling a ranking ρ and an adjacent-rank transposition σ from $\mathcal{D}_T$, in particular via SAMPLE–$\mathcal{D}_T$. For any fixed execution up through Line 4, we study the effect of the randomness in Line 5 in regards to the condition (4.39) from Lemma 4.44. After Line 5, it is determined whether (4.39) holds.

Since we have $n_{v,1} = 1$, either A or B is empty, and the other has size $n_{v,2} - 1$. Thus $\delta(A, B)$ is zero, and

$$-|A_<| + |A_>| + |B_<| - |B_>| = (n_{v,2} - 1) - 2k \tag{4.67}$$

for some integer k with $0 \le k \le n_{v,2} - 1$ that is determined by Line 5. In particular, k is uniformly random on all the integers in that range. Since there is at most one value of k such that $(n_{v,2} - 1) - 2k = 0$, we conclude via Lemma 4.44 that $\hat{s}_{T,v} \ge 1 - 1/n_{v,2}$. ∎

In the case $\min(n_{v,1}, n_{v,2}) \ge 2$, we approach (4.39) from a different direction. Whereas previously we made use of the randomness in the right-hand side when the left-hand side is fixed, we instead make use of the randomness in the left-hand side while the right-hand side is fixed. In order to do this effectively, it is useful to expand on (4.39). The following lemma formulates this.

Lemma 4.55. *Let T be a rooted tree, and let v be a node in T such that v has exactly two children and such that the subtree rooted at v is not freely arrangeable. For any ranking ρ*

of the leaves of T and any adjacent-rank transposition σ, we have $X_{T,v}(\rho) \neq X_{T,v}(\sigma\rho)$ if and only if

$$\delta_\rho(A_< : B_<) + \delta_\rho(A_> : B_>) \neq -|A_<||B_>| + |A_>||B_<| - |A_<| + |A_>| + |B_<| - |B_>|, \tag{4.68}$$

where the quantities in (4.68) are those defined in Definition 4.43.

Proof. This follows from Lemma 4.44 and the identity

$$\delta_\rho(A : B) = \delta_\rho(A_< : B_<) + |A_<||B_>| - |A_>||B_<| + \delta_\rho(A_> : B_>). \tag{4.69}$$

■

Now, recall SAMPLE–$\mathcal{D}_T$ (Algorithm 3) from Subsection 4.6.2. The condition (4.68) is determined upon completion of Line 5. The elements of A and B that precede the $*$ in R respectively constitute $A_<$ and $B_<$, and $\delta_\rho(A_< : B_<)$ is determined by how those elements are interleaved; similarly, $A_>$, $B_>$, and $\delta_\rho(A_> : B_>)$ are determined by the contents of R following the $*$. In particular, conditioned on $A_<$ and $B_<$, $\delta_\rho(A_< : B_<)$ is distributed identically to $\delta_{\rho'}(A_< : B_<)$ when ρ' is a uniformly random ranking of $A_< \sqcup B_<$, and, similarly, we know the distribution of $\delta_\rho(A_> : B_>)$ conditioned on $A_>$ and $B_>$. Moreover, they are independent. Thus, conditioned on $A_<$, $A_>$, $B_<$, and $B_>$, $\delta_\rho(A_< : B_<) + \delta_\rho(A_> : B_>)$ is distributed identically to $\delta_{\rho'}(A_< : B_<) + \delta_{\rho''}(A_> : B_>)$ where ρ' and ρ'' are independent uniformly random rankings of $A_< \sqcup B_<$ and $A_> \sqcup B_>$, respectively. In this way, we can fix the right-hand side of (4.68) and leave the left-hand side comprehensibly random.

This leads to the following analysis for $\hat{s}_{T,v}$. As above, we condition on $A_<$, $A_>$, $B_<$, and $B_>$, and consider the left-hand side of (4.68). If we are in a case where both $A_<$ and $B_<$ both make up at least an η-fraction of A and B, respectively, then the probability that $\delta_\rho(A_< : B_<)$ takes on any particular value is bounded by $\gamma_\eta(|A|, |B|) = \gamma_\eta(n_{v,1} - 1, n_{v,2} - 1)$. Since $\delta_\rho(A_< : B_<)$ and $\delta_\rho(A_> : B_>)$ are independent, the same bound carries over to their sum, and Lemma 4.54 follows in this case. Similarly, Lemma 4.54 follows if both $A_>$ and $B_>$ make up at least an η-fraction of A and B.

Since $A_<$ and $A_>$ partition A, one or the other contains an η-fraction of A for any $\eta \le 1/2$. The same holds with respect to $B_<$, $B_>$, and B. It follows that one of the preceding cases applies, except for when either both $A_<$ and $B_>$ are too small, or else both $A_>$ and $B_<$ are too small. In these cases, however, the complementary sets are large, and we take another look at (4.68): on the right-hand side, we have a term, $|A_>||B_<|$ or $|A_>||B_<|$, respectively, that is large in magnitude compared to all the others, even the terms on the left-hand side. In particular, when η is a small enough constant, we can show that the right-hand side of (4.68) always exceeds the left-hand side in absolute value. Thus in the exceptional cases, $X_{T,v}$ is in fact sensitive with probability 1.

Here is the formal proof.

Proof of Lemma 4.54, when $n_{v,1} \ge 2$ and $n_{v,2} \ge 2$. Let v be as in the lemma statement, and let η be a sufficiently small constant to be determined later. We consider sampling from $\mathcal{D}_T$ conditioned on $\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}}) = v$ and that ℓ_{lo} and ℓ_{hi} are not coisolated. Recalling the random variables in Definition 4.43, we let E denote the collective outcome for ℓ_{lo}, ℓ_{hi}, A, $A_<$, $A_>$, B, $B_<$, and $B_>$. Using Lemma 4.55, we write

$$\hat{s}_{T,v} = 1 - \sum_{E_0} \Pr[E = E_0] \Pr\left[\delta_\rho\left(A_< : B_<\right) + \delta_\rho\left(A_> : B_>\right) = k_0 \mid E = E_0\right] \tag{4.70}$$

$$\ge 1 - \max_{E_0} \Pr\left[\delta_\rho\left(A_< : B_<\right) + \delta_\rho\left(A_> : B_>\right) = k_0 \mid E = E_0\right] \tag{4.71}$$

where E_0 ranges over the possible outcomes for E, and

$$k_0 \doteq -|A_<||B_>| + |A_>||B_<| - |A_<| + |A_>| + |B_<| - |B_>| \tag{4.72}$$

is the right-hand side of (4.68). Let $q(E_0)$ be the probability in the max in (4.71). We bound (4.71) below by separately bounding each $q(E_0)$ above.

We begin by considering the case where $|A_<| \ge \eta|A|$ and $|B_<| \ge \eta|B|$. As mentioned in the discussion preceding this proof, when we condition on $E = E_0$, $\delta_\rho\left(A_< : B_<\right)$ is independent of $\delta_\rho\left(A_> : B_>\right)$, and the probability that $\delta_\rho\left(A_< : B_<\right)$ takes on the value k is $g(|A_<|, |B_<|, k)$.

Thus we have

$$q(E_0) = \sum_k \Pr[\delta_\rho(A_< : B_<) = k \text{ and } \delta_\rho(A_> : B_>) = k_0 - k \mid E = E_0] \tag{4.73}$$

$$= \sum_k g(|A_<|, |B_<|, k) \cdot \Pr[\delta_\rho(A_> : B_>) = k_0 - k \mid E = E_0] \tag{4.74}$$

$$\leq \max_k g(|A_<|, |B_<|, k) \tag{4.75}$$

$$\leq \gamma_\eta(|A|, |B|). \tag{4.76}$$

The case where $|A_>| \geq \eta|A|$ and $|B_>| \geq \eta|B|$ is symmetric, and concludes with the same bound.

Next, we consider the case where $|A_<| \leq \eta|A|$ and $|B_>| \leq \eta|B|$. Writing $|A_<| = \eta_A|A|$ and $|B_>| = \eta_B|B|$, we have $\eta_A \leq \eta$ and $\eta_B \leq \eta$. Under the hypothesis that $\eta \leq 1/2$, we have

$$|\delta_\rho(A_< : B_<) + \delta_\rho(A_> : B_>)| \leq |A_<||B_<| + |A_>||B_>| \tag{4.77}$$

$$= (\eta_A(1-\eta_B) + (1-\eta_A)\eta_B)|A||B| \tag{4.78}$$

$$\leq 2\eta(1-\eta)|A||B|. \tag{4.79}$$

There is also the bound

$$k_0 \geq -\eta^2|A||B| + (1-\eta)^2|A||B| - \eta|A| + (1-\eta)|A| + (1-\eta)|B| - \eta|B| \tag{4.80}$$

$$= (1-2\eta) \cdot (|A||B| + |A| + |B|). \tag{4.81}$$

Since $\{|A|, |B|\} = \{n_{v,1} - 1, n_{v,2} - 1\}$ and $\min(n_{v,1}, n_{v,2}) \geq 2$, both $|A|$ and $|B|$ are strictly positive. It follows that, for any η such that $2\eta(1-\eta) \leq (1-2\eta)$, i.e., for any $\eta \in (0, 1 - \frac{1}{2}\sqrt{2}]$, we have the strict inequality $\delta_\rho(A_< : B_<) + \delta_\rho(A_> : B_>) < k_0$. Consequently, in this case, $q(E_0) = 0$. Similarly, in the case where $|A_>| \leq \eta|A|$ and $|B_<| \leq \eta|B|$, k_0 is bounded above by $-(1-2\eta)(|A||B| + |A| + |B|)$, so $\delta_\rho(A_< : B_<) + \delta_\rho(A_> : B_>) > k_0$, and again we have $q(E_0) = 0$.

Finally, as long as $\eta \leq 1/2$, it holds that either $|A_<| \geq \eta|A|$ or $|A_>| \geq \eta|A|$, and either $|B_<| \geq \eta|B|$ or $|B_>| \geq \eta|B|$. Distributing the "and" over the "or", we obtain the four cases just discussed. In each case we have $q(E_0) \leq \gamma_\eta(n_{v,1} - 1, n_{v,2} - 1)$, provided η satisfies all the requisite bounds. The most stringent is $\eta \leq 1 - \frac{1}{2}\sqrt{2} \approx 0.29289$. Lemma 4.54 (which uses $\eta = 1/4$ for simplicity) now follows from (4.71). ∎

4.7.3 A Conjecture on the Gaussian Binomial Coefficients

Finally, we discuss the conjecture that underlies Theorems 4.47 and 4.48. It concerns the so-called Gaussian binomial coefficients.

Definition 4.56. For natural numbers a, b, k, the (a, b, k)-th Gaussian binomial coefficient, denoted by $G(a, b, k)$, is the coefficient of q^k in the polynomial

$$\psi_{a,b}(q) \doteq \frac{\phi_{a+b}(q)}{\phi_a(q)\phi_b(q)} \tag{4.82}$$

where

$$\phi_k(q) = (1)(1+q^1)(1+q^1+q^2)\cdots(1+q^1+\cdots+q^{k-1}) \tag{4.83}$$

with $\phi_0(q) = \phi_1(q) = 1$. ◂

Each $\psi_{a,b}$ is indeed a polynomial as can be seen by observing that the family satisfies the following recurrence:

$$\psi_{a,b}(q) = \begin{cases} 1 & \text{if } a = 0 \text{ or } b = 0 \\ \psi_{a-1,b}(q) + q^a \cdot \psi_{a,b-1}(q) & \text{if } a > 0 \text{ and } b > 0 \end{cases} \tag{4.84}$$

The recurrence can also be used to show that $G(a, b, k)$ is zero except for $k = 0, \ldots, ab$, where $G(a, b, k)$ is a positive integer.

One interpretation of $G(a, b, k)$ is that it counts the number of ways to walk from the origin to (a, b) with steps $(0, 1)$ or $(1, 0)$, and with an area of k between the path taken (treating each step as a line segment) and the horizontal axis. Another understanding—more relevant to this chapter—is that they count the number of rankings with a fixed number of inversions.

Fact 4.57. *Let a and b be non-negative integers, and let k be an integer with $0 \le k \le ab$. Then $G(a, b, k)$ counts the number of rankings $\rho : [a + b] \to [a + b]$ with the following properties:*

- $\rho(1) < \rho(2) < \cdots < \rho(a)$,

- $\rho(a+1) < \rho(a+2) < \cdots < \rho(a+b)$, *and*
- *The number of pairs* $i, j \in [a+b]$ *with* $i < j$ *but* $\rho(i) > \rho(j)$ *is exactly* k.

For fixed a, b, the sum over k of $G(a,b,k)$ is $\binom{a+b}{a}$. Thus the quantities $G(a,b,k)/\binom{a+b}{a}$ determine a probability distribution over choices of k. Indeed, we have

$$\frac{G(a,b,k)}{\binom{a+b}{a}} = g(a,b,ab-2k), \tag{4.85}$$

where $g(a,b,ab-2k)$ is the quantity in Definition 4.52. We are interested in an upper bound on the probability of any single outcome, independent of k. If $a = 0$ or $b = 0$, then there are no nontrivial bounds, but for our purposes it suffices to study the case where both a and b are positive.

It can be verified from (4.84) that the distribution has mean $ab/2$ and variance $\frac{1}{12}ab(a+b+1)$. It is known that for any sequence of choices of (a,b) in which both a and b grow unboundedly, the limiting distribution is a normal distribution [MW47; Tak86]. Thus "in the limit" $G(a,b,k)$ coincides with the probability that a normal distribution with the same mean and variance takes a value near k. The maximum density of that distribution is

$$\sqrt{\frac{1}{2\pi}}\sqrt{\frac{12}{ab(a+b+1)}}, \tag{4.86}$$

and we expect that to be approximately an upper bound when a and b are large. Meanwhile, in the extremal case where $a = 1$, we have $G(a,b,k) = 1$ for all $k = 0, \ldots, b$, in which case the distribution is uniform with maximum probability $1/(b+1)$. Again, (4.86) is an upper bound, up to a constant factor. The following conjecture asserts that in fact (4.86) holds up to a constant factor for all $a, b \geq 1$.

Conjecture 4.58. *There is a constant* c *so that, for all* $a, b \geq 1$, *and all* $0 \leq k \leq ab$,

$$G(a,b,k) \leq \frac{c}{\sqrt{ab(a+b+1)}}\binom{a+b}{a}. \tag{4.87}$$

Existing results suffice to prove this conjecture under the additional hypothesis that $\min(a,b) \leq L$ (where c depends L) [SZ16], as well as separately under the hypothesis that

$\min(a,b) \geq \alpha \cdot (a+b)$ for some $\alpha > 0$ (where c depends on α) [MPP20]. All together, the evidence points toward a positive resolution of Conjecture 4.58.

We used Conjecture 4.58 and its proven special cases in Subsection 4.7.2 to bound the quantities $\gamma_{1/4}(n_{v,1}-1, n_{v,2}-1)$ in Lemma 4.54. We conclude with the proofs of the two bounds.

Fact 4.59. *Assuming Conjecture 4.58, for every $\eta \in (0,1)$, there is a constant $c > 0$ such that for every $\hat{a}, \hat{b} \geq 1$, we have*

$$\gamma_\eta\left(\hat{a},\hat{b}\right) \leq \frac{c}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}}. \tag{4.88}$$

Proof of Fact 4.59. Fix integers a, b such that $\eta\hat{a} \leq a \leq \hat{a}$ and $\eta\hat{b} \leq b \leq \hat{b}$. Since a and b are integers and $\eta\hat{a}$ and $\eta\hat{b}$ are positive, we have $a \geq 1$ and $b \geq 1$. It follows that $a+1 \leq 2a$, $b+1 \leq 2b$, and $2+a+b \leq 2(a+b+1)$. We also have $(1+a)/\eta \geq 1+\hat{a}$, $(1+b)/\eta \geq 1+\hat{b}$, and $(2+a+b)/\eta \geq (2+\hat{a}+\hat{b})$. Let c_0 be the constant guaranteed by Conjecture 4.58. Then we have the following chain of inequalities.

$$G(a,b,r) \leq \frac{c_0}{\sqrt{ab(a+b+1)}} \cdot \binom{a+b}{a} \tag{4.89}$$

$$\leq \frac{2\sqrt{2}c_0}{\sqrt{(1+a)(1+b)(2+a+b)}} \cdot \binom{a+b}{a} \tag{4.90}$$

$$\leq \frac{2\sqrt{2}c_0/\eta^{3/2}}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}} \cdot \binom{a+b}{a} \tag{4.91}$$

Taking $c = 2\sqrt{2}c_0/\eta^{3/2}$ completes the proof. ∎

Theorem 4.60. *For any constants $\alpha > 0$, $L \geq 1$, and $\eta > 0$, there is a constant c such that the following holds. For any $\hat{a}, \hat{b} \geq 1$ such that either*

- $\min(1+\hat{a}, 1+\hat{b}) \leq L$, *or*
- $\min(1+\hat{a}, 1+\hat{b}) \geq \alpha \cdot (2+\hat{a}+\hat{b})$,

we have

$$\gamma_\eta\left(\hat{a},\hat{b}\right) \le \frac{c}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}}. \tag{4.92}$$

Proof of Theorem 4.60. Fix α, L, η, $\hat{a}$, and $\hat{b}$ as in the theorem statement.

We first consider the case where $\min(1+\hat{a}, 1+\hat{b}) \le L$. For every a and b with $\eta\hat{a} \le a \le \hat{a}$ and $\eta\hat{b} \le b \le \hat{b}$, we have $\min(a,b) \le \min(\hat{a},\hat{b}) \le L-1$. Given that, Theorem 2.4 in [SZ16] states that there is a constant c_1, depending on L, so that the following holds.

$$G(a,b,r) \le \frac{c_1}{\max(a,b)} \cdot \binom{a+b}{a} \tag{4.93}$$

Since we have $a \ge \eta\hat{a}$, $b \ge \eta\hat{b}$, $\hat{a} \ge 1$, and $\hat{b} \ge 1$, we may write

$$\begin{aligned}
\max(a,b) &\ge \eta \max(\hat{a},\hat{b}) && (4.94)\\
&\ge \frac{\eta}{2}\max(1+\hat{a}, 1+\hat{b}) && (4.95)\\
&\ge \frac{\eta}{2\sqrt{2}}\sqrt{\max(1+\hat{a},1+\hat{b})}\cdot\sqrt{2+\hat{a}+\hat{b}} && (4.96)
\end{aligned}$$

where the last inequality uses that the maximum of two numbers is at least their average. Plugging into (4.93) we obtain

$$\begin{aligned}
G(a,b,r) &\le \frac{2\sqrt{2}c_1\sqrt{\min(1+\hat{a},1+\hat{b})}/\eta}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}} \cdot \binom{a+b}{a} && (4.97)\\
&\le \frac{2\sqrt{2}c_1\sqrt{L}/\eta}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}} \cdot \binom{a+b}{a}. && (4.98)
\end{aligned}$$

Thus the theorem holds true in this case as long as $c \ge 2\sqrt{2}c_1\sqrt{L}/\eta$.

Next we consider the case where $\min(\hat{a},\hat{b}) \ge \alpha\cdot(\hat{a}+\hat{b})$. First, since $\min(1+\hat{a},1+\hat{b}) \ge 2$, we assume without loss of generality that $\alpha \ge 2/(\hat{a}+\hat{b}+2) \ge 1/(\hat{a}+\hat{b})$. We also assume without loss of generality that $\alpha \le 1/2$. With that in mind, we set $\alpha' \doteq 2\alpha^2$, and derive

$$\begin{aligned}
\min(\hat{a},\hat{b}) &\ge \alpha\cdot(\hat{a}+\hat{b}+2) - 1 && (4.99)\\
&= \alpha'\cdot(\hat{a}+\hat{b}) + \left(\alpha(\hat{a}+\hat{b})-1\right)(1-2\alpha) && (4.100)\\
&\ge \alpha'(\hat{a}+\hat{b}). && (4.101)
\end{aligned}$$

Now, for every a and b with $\eta\hat{a} \le a \le \hat{a}$ and $\eta\hat{b} \le b \le \hat{b}$, it holds that

$$\min(a,b) \ge \eta \min(\hat{a},\hat{b}) \ge \alpha'\eta(\hat{a}+\hat{b}) \ge \alpha'\eta(a+b). \tag{4.102}$$

Given that, Theorem 1 from [MPP20] implies that there is a constant c_2, depending on $\alpha'\eta$ (hence only on α and η), so that

$$G(a,b,r) \le \frac{c_2}{ab} \cdot \frac{(a+b)^{a+b}}{a^a b^b}. \tag{4.103}$$

From Stirling's approximation, there is a constant c_3 so that

$$\frac{(a+b)^{a+b}}{a^a b^b} \le c_3 \sqrt{\frac{ab}{a+b}} \cdot \binom{a+b}{a}. \tag{4.104}$$

Together with (4.103), we have

$$G(a,b,r) \le \frac{c_2 c_3}{\sqrt{ab(a+b)}} \cdot \binom{a+b}{a}. \tag{4.105}$$

Since $a \ge \eta\hat{a}$, $b \ge \eta\hat{b}$, $\hat{a} \ge 1$, and $\hat{b} \ge 1$, we obtain

$$G(a,b,r) \le \frac{c_2 c_3/\eta^{3/2}}{\sqrt{\hat{a}\hat{b}(\hat{a}+\hat{b})}} \cdot \binom{a+b}{a} \tag{4.106}$$

$$\le \frac{2\sqrt{2}c_2 c_3/\eta^{3/2}}{\sqrt{(1+\hat{a})(1+\hat{b})(2+\hat{a}+\hat{b})}} \cdot \binom{a+b}{a}. \tag{4.107}$$

The theorem thus holds in this case as long as $c \ge 2\sqrt{2}c_2 c_3/\eta^{3/2}$.

Taking $c = \max(2\sqrt{2}c_1\sqrt{L}/\eta, 2\sqrt{2}c_2 c_3/\eta^{3/2})$ satisfies both cases, proving the theorem. ■

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.

[AD17] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Information and Computation*, 256:2–8, 2017.

[AFS+18] Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. *ACM Transactions on Computation Theory*, 10(1):1–30, 2018.

[AGK+15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015.

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 92–105, 2005.

[AGvM+18a] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. In *Innovations in Theoretical Computer Science (ITCS)*, 20:1–20:20, 2018.

[AGvM+18b] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM Journal on Computing*, 47(4):1339–1372, 2018.

[AK06] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Information and Computation*, 204(5):835–852, 2006.

[AKR+10] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77:14–40, 2010.

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *ACM Symposium on Theory of Computing (STOC)*, pages 321–330, 2013.

[AvMV15] Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Computational Complexity*, 24(4):695–776, 2015.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time. In *ACM Symposium on Theory of Computing (STOC)*, pages 684–697, 2016.

[Bab91] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *ACM Symposium on Theory of Computing (STOC)*, pages 164–174, 1991.

[BBS09] László Babai, Robert Beals, and Ákos Seress. Polynomial-time theory of matrix groups. In *ACM Symposium on Theory of Computing (STOC)*, pages 55–64, 2009.

[BFP+73] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[BG21] Vishwas Bhargava and Sumanta Ghosh. Improved hitting set for orbit of ROABPs. In *International Conference on Randomization and Computation (RANDOM)*, 30:1–30:23, 2021.

[BG84a] Andreas Blass and Yuri Gurevich. Equivalence relations, invariants, and normal forms. *SIAM Journal on Computing*, 13(4):682–689, 1984.

[BG84b] Andreas Blass and Yuri Gurevich. Equivalence relations, invariants, and normal forms, II. In *Logic and Machines: Decision Problems and Complexity*, volume 171 of *Lecture Notes in Computer Science*, pages 24–42, 1984.

[CFJ+10] Jean Cardinal, Samuel Fiorini, Gwenaël Joret, Raphaël M. Jungers, and J. Ian Munro. An efficient algorithm for partial order production. *SIAM Journal on Computing*, 39(7):2927–2940, 2010.

[CIK+16] Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Computational Complexity Conference (CCC)*, 10:1–10:24, 2016.

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

[CW79] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[Deg82] Richard Degerman. Ordered binary trees constructed through an application of Kendall's tau. *Psychometrika*, 47(4):523–527, 1982.

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[DZ01] Dorit Dor and Uri Zwick. Median selection requires $(2 + \epsilon)n$ comparisons. *SIAM Journal on Discrete Mathematics*, 14(3):312–325, 2001.

[DZ99] Dorit Dor and Uri Zwick. Selecting the median. *SIAM Journal on Computing*, 28(5):1722–1758, 1999.

[ER65] Paul Erdős and Alfréd Rényi. Probabilistic methods in group theory. *Journal d'Analyse Mathématique*, 14(1):127–138, 1965.

[FG11] Lance Fortnow and Joshua A. Grochow. Complexity classes of equivalence problems revisited. *Information and Computation*, 209(4):748–763, 2011.

[FH16] Jeffrey Finkelstein and Benjamin Hescott. Polynomial-time kernel reductions, 2016. arXiv: `1604.08558 [cs.CC]`.

[FK18] Hervé Fournier and Arpita Korwar. Limitations of the Shpilka–Volkovich generator. Workshop on Algebraic Complexity Theory (WACT), Paris, 2018.

[For15] Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 451–465, 2015.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 243–252, 2013.

[FSS13] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order, 2013. arXiv: 1309.5668 [cs.CC]. Full version of [FSS14].

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *ACM Symposium on Theory of Computing (STOC)*, pages 867–875, 2014.

[FSV17] Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct hitting sets and barriers to proving algebraic circuits lower bounds. In *ACM Symposium on Theory of Computing (STOC)*, pages 653–664, 2017.

[FSV18] Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct hitting sets and barriers to proving algebraic circuits lower bounds, 2018. arXiv: 1701.05328 [cs.CC]. Full version of [FSV17].

[GG20] Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for ROABPs. In *International Conference on Randomization and Computation (RANDOM)*, 2020.

[GK00] Hermann Grassmann and Lloyd C Kannenberg. *Extension theory*. American Mathematical Society, 2000. Translation of [Gra44].

[GKS+17] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *Computational Complexity*, 26(4):835–880, 2017.

[GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[GQ21] Joshua A. Grochow and Youming Qiao. On the complexity of isomorphism problems for tensors, groups, and polynomials I: tensor isomorphism-completeness. In *Innovations in Theoretical Computer Science (ITCS)*, 31:1–31:19, 2021.

[Gra44] Hermann Grassmann. *Die lineale Ausdehnungslehre ein neuer Zweig der Mathematik: dargestellt und durch Anwendungen auf die übrigen Zweige der Mathematik, wie auch auf die Statik, Mechanik, die Lehre vom Magnetismus und die Krystallonomie erläutert*. O. Wigand, 1844.

[Gro12] Joshua A. Grochow. Matrix Lie algebra isomorphism. In *Annual IEEE Conference on Computational Complexity (CCC)*, pages 203–213, 2012.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of Computational Group Theory*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 2005.

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute. In *ACM Symposium on Theory of Computing (STOC)*, pages 262–272, 1980.

[IW97] Russell Impagliazzo and Avi Wigderson. P=BPP if E requires exponential circuits: derandomizing the XOR lemma. In *ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

[JQS09] Maurice Jansen, Youming Qiao, and Jayalal Sarma. Deterministic identity testing of read-once algebraic branching programs, 2009. arXiv: `0912.2565 [cs.CC]`.

[JQS10] Maurice Jansen, Youming Qiao, and Jayalal Sarma M. N. Deterministic black-box identity testing π-ordered algebraic branching programs. In *IARCS Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 296–307, 2010.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1–2):1–46, 2004.

[KMM+05] Kanela Kaligosi, Kurt Mehlhorn, J. Ian Munro, and Peter Sanders. Towards optimal multiple selection. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 103–114, 2005.

[KMS+13] Zohar S. Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in. *SIAM Journal on Computing*, 42(6):2114–2131, 2013.

[Knu98] Donald E. Knuth. *The Art of Computer Programming. Volume 3: Sorting and Searching*. Addison-Wesley, 2nd edition, 1998.

[Kor21] Arpita Korwar. Personal communication, 2021.

[KS17] Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. *Theory of Computing*, 13(1):1–33, 2017.

[KST93] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser Verlag, 1993.

[MPP20] Stephen Melczer, Greta Panova, and Robin Pemantle. Counting partitions inside a rectangle. *SIAM Journal on Discrete Mathematics*, 34(4):2388–2410, 2020.

[MS21] Dori Medini and Amir Shpilka. Hitting sets and reconstruction for dense orbits in VP_e and $\Sigma\Pi\Sigma$ circuits. In *Computational Complexity Conference (CCC)*, 19:1–19:27, 2021.

[MV18] Daniel Minahan and Ilya Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *ACM Transactions on Computation Theory (TOCT)*, 10(3):1–11, 2018.

[MW47] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *ACM Symposium on Theory of Computing (STOC)*, pages 410–418, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[ODo14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[Ore22] Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7), 1922.

[PP10] Ramamohan Paturi and Pavel Pudlák. On the complexity of circuit satisfiability. In *ACM Symposium on Theory of Computing (STOC)*, pages 241–250, 2010.

[PR97] Erez Petrank and Ron M. Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, 1997.

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

[Rud17] M. Rudow. Discrete logarithm and minimum circuit size. *Information Processing Letters*, 128:1–4, 2017.

[Sch76] Arnold Schönhage. The production of partial orders. *Astérisque*, 38–39:229–246, 1976.

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

[Ser03] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.

[Ser21] I. S. Sergeev. On the upper bound of the complexity of sorting. *Computational Mathematics and Mathematical Physics*, 61(2):329–346, 2021.

[ST21] Chandan Saha and Bhargav Thankey. Hitting sets for orbits of circuit classes and polynomial families. In *International Conference on Randomization and Computation (RANDOM)*, 50:1–50:26, 2021.

[SV15] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015.

[SY10] Amir Shpilka and Amir Yehudayoff. *Arithmetic circuits: A survey of recent results and open questions*. Now Publishers Inc, 2010.

[SZ16] Richard P. Stanley and Fabrizio Zanello. Some asymptotic results on q-binomial coefficients. *Annals of Combinatorics*, 20(3):623–634, 2016.

[Tak86] Lajos Takács. Some asymptotic formulas for lattice paths. *Journal of Statistical Planning and Inference*, 14(1):123–142, 1986.

[Tra84] Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

[vMM22] Dieter van Melkebeek and Andrew Morgan. Polynomial identity testing via evaluation of rational functions. In *Innovations in Theoretical Computer Science (ITCS)*, 2022. To appear.

[Wie20] Daniel Wiebking. Normalizers and permutational isomorphisms in simply-exponential time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 230–238, 2020.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226, 1979.