COSC 544 Probabilistic Proof Systems 9/5/17

Interactive Proofs & Arguments, Low-Degree & Multilinear Extensions

Lecturer: Justin Thaler

1 Definitions: Interactive Proofs and Argument Systems

Throughout these notes, boldface will be used for vectors, to distinguish them from scalars. The notation $\tilde{O}(\cdot)$ hides polylogarithmic factors. So, for example, $n \log^4 n = \tilde{O}(n)$.

1.1 Interactive Proofs

Definition 1.1. Given a function f mapping $\{0,1\}^n$ to a finite range $\mathcal{R}$, an *interactive proof system* for f consists of a probabilistic polynomial time verifier $\mathcal{V}$ and a prover $\mathcal{P}$ who are given a common input $\mathbf{x} \in \{0,1\}^n$. $\mathcal{P}$ and $\mathcal{V}$ exchange a sequence of messages to produce a transcript $\mathbf{t} = (\mathcal{V}(\mathbf{r}), \mathcal{P})(\mathbf{x})$, where $\mathbf{r}$ denotes $\mathcal{V}$'s internal randomness. After the transcript $\mathbf{t}$ is produced, $\mathcal{V}$ must output value in $\mathcal{R} \cup \{\perp\}$, where $\perp$ is a special "rejection" symbol indicating that $\mathcal{V}$ rejects $\mathcal{P}$'s claims as invalid. Denote by $\text{out}(\mathcal{V}, \mathbf{x}, \mathbf{r}, \mathcal{P})$ the output of verifier $\mathcal{V}$ on input $\mathbf{x}$ given prover strategy $\mathcal{P}$ and that $\mathcal{V}$'s internal randomness is equal to $\mathbf{r}$.

The interactive proof system has completeness error δ_c and soundness error δ_s if the following two properties hold.

1. *(Completeness)* There exists a prover strategy $\mathcal{P}$ such that for every $\mathbf{x} \in \mathcal{L}$,

$$\Pr[\text{out}(\mathcal{V}, \mathbf{x}, \mathbf{r}, \mathcal{P}) = f(\mathbf{x})] \geq 1 - \delta_c.$$

2. *(Soundness)* For every $\mathbf{x} \notin \mathcal{L}$ and every prover strategy $\mathcal{P}'$,

$$\Pr[\text{out}(\mathcal{V}, \mathbf{x}, \mathbf{r}, \mathcal{P}') \notin f(\mathbf{x}), \{\perp\}] \leq \delta_s.$$

An interactive proof system is valid if $\delta_c, \delta_s \leq 1/3$. The complexity class **IP** is the class of all languages possessing valid interactive proof systems.

Intuitively, for any input $\mathbf{x}$, the completeness condition requires that there be a convincing proof for what is the value of f at $\mathbf{x}$. The soundness condition requires that false statements of the form "$f(\mathbf{x}) = z$" for any $z \neq f(\mathbf{x})$ lack a convincing proof.

Several additional clarifying remarks are in order.

- All of the interactive proofs that we will see in this course actually satisfy *perfect* completeness, meaning that $\delta_c = 0$. That is, the honest prover will *always* convince the verifier that it is honest.
- Regarding the requirement that $\delta_s \leq 1/3$, the constant $1/3$ is chosen by convention. In all of the interactive proofs that we see in this course, the soundness error will always be proportional to $1/|\mathbb{F}|$, where $\mathbb{F}$ is the field over which the interactive proof is defined. In practice, $1/|\mathbb{F}|$ be astronomically small (e.g. smaller than, say, 2^{-60}).

- We highlight the fact that the soundness requirement in Definition 1.1 is required to hold even against computationally unbounded provers P', who might be devoting enormous computational resources to trying to trick $\mathcal{V}$ into outputting an incorrect answer.
- Observe that Definition 1.1 implicitly assumes that the total number of messages exchanged by $\mathcal{P}$ and $\mathcal{V}$ is $\text{poly}(n)$, as the definition requires that $\mathcal{V}$ run in $\text{poly}(n)$ time over the entire course of the interaction (if the number of messages were superpolynomial in n, then $\mathcal{V}$ could not even read all of the messages in $\text{poly}(n)$ time).
- We clarify that in an interactive proof system, $\mathcal{V}$'s randomness is internal, and in particular is not visible to the prover.

The two costs of paramount importance in any interactive proof are $\mathcal{P}$'s runtime and $\mathcal{V}$'s runtime, but there are other important costs as well: $\mathcal{P}$ and $\mathcal{V}$'s space usage, the total number of bits communicated, and the total number of messages exchanged. If $\mathcal{V}$ and $\mathcal{P}$ exchange at most m messages for every pair $(\mathbf{x}, \mathbf{r})$, then $\lceil m/2 \rceil$ is referred to as the *round complexity* of the interactive proof system.

Robustness of the Model. The key to the power of interactive proofs is the combination of randomness and interaction. If no interaction is allowed, but the verifier is allowed to toss random coins and accept an incorrect proof with small probability, the resulting complexity class is known as **MA**. This class is widely believed to be equal to **NP**, a much smaller class than **PSPACE** = **IP**. Meanwhile, if no randomness is allowed (equivalently, if perfect soundness is required), then the resulting complexity class is once again **IP**.

However, as long as we allow both randomness and interaction, the interactive proofs model is robust to a wide variety of tweaks to the definition.

- (Public Coins vs. Private Coins). Interactive proofs were introduced in 1985 by Goldwasser, Micali, and Rackoff [GMR89]. At the same conference, Babai [Bab85] independently introduced the Arthur-Merlin class hierarchy, which captures constant-round interactive proof systems, with the additional requirement that the verifier's randomness is public—that is, visible to the prover. Goldwasser and Sipser [GS86] subsequently proved that the distinction between public and private coins is not crucial: any constant-round private coin interactive proof system can be simulated by a constant-round public coin system (with a polynomial blowup in costs).
- (2 Rounds vs. $O(1)$ Rounds). Babai and Moran showed that any constant-round interactive proof can be simulated by a 2-message interactive proof with a polynomial blowup in costs [Bab85, BM88].
- (Perfect vs. Imperfect Completeness). Goldwasser and Sipser [GS86] also showed that any interactive proof with imperfect completeness can be simulated by an interactive proof with perfect completeness.

On Interactive Proofs for Languages Versus Functions. A *language* $\mathcal{L} \subseteq \{0,1\}^*$ is any subset of Boolean strings (here, $\{0,1\}^*$ denotes the set of all finite Boolean strings). In an interactive proof for $\mathcal{L}$, the verifier $\mathcal{V}$ interacts with a prover $\mathcal{P}$ in exactly the same manner as in Definition 1.1, and at the end of the interaction, $\mathcal{V}$ must either output "accept" or "reject". The requirements are:

- **Completeness**. For any $\mathbf{x} \in \mathcal{L}$, there is some prover strategy will cause the verifier to accept with high probability.
- **Soundness**. For any $\mathbf{x} \notin \mathcal{L}$, then for *every* prover strategy, the verifier outputs reject with high probability.

Note in particular that, for $\mathbf{x} \notin \mathcal{L}$, it is *not* required that there to a "convincing proof" of the fact that $f_{\mathcal{L}}(\mathbf{x}) = 0$. The reasoning behind this formalization of interactive proofs for languages is as follows. One thinks of inputs in the language as *true statements*, and inputs not in the language as *false statements*. The above completeness and soundness properties require that all true statements have convincing proofs, and all false statements do not have convincing proofs. It is *not* required that false statements have convincing refutations (i.e., convincing proofs of their falsity).

In contrast, Definition 1.1 (which defines an interactive proof system for a *function f*) requires that for *any* $\mathbf{x}$, there is a convincing proof of the value of $f(\mathbf{x})$. The notions of interactive proofs for languages and functions are, however, related in the following sense: given a *function f*, an interactive proof for f is equivalent to an interactive proof for the *language $\mathcal{L}_f := \{(\mathbf{x}, y) : y = f(\mathbf{x})\}$*.

In these lecture notes, we will primarily be concerned with interactive proofs for functions instead of languages (we only talk about interactive proofs for languages when referring to complexity classes such as **IP**, defined in the next subsection).

1.1.1 NP and IP

Let **IP** be the class of all languages solvable by an interactive proof system (with a polynomial time verifier). The class **IP** can be viewed as an interactive, randomized variant of the classical complexity class **NP** (**NP** is the class obtained by restricting the proof system to be non-interactive and deterministic, meaning that the completeness and soundness errors are 0).

We will see soon that the class **IP** is in fact equal to **PSPACE**, the class of all languages solvable by algorithms using polynomial space (and possibly exponential time). **PSPACE** is believed to be a vastly bigger class of languages than **NP**, so this is one formalization of the statement that "interactive proofs are far more powerful than classical static (i.e, **NP**) proofs".

Interestingly, *both* ingredients (interaction and randomness) seem necessary to make the resulting proof systems substantially more powerful than static proofs. In particular, if we allow proof systems to be interactive but not randomized, then the class of languages solvable by such proof systems with polynomial time verifiers is still just **NP** (showing this may be a question on the first problem set). And if we allow proof systems to be randomized by not interactive, then the class of languages solvable by such proof systems with polynomial time verifiers is called **MA** (short for Merlin-Arthur). Many people believe that **MA** = **NP**; that is, it is suspected that allowing randomness but not interaction in proof systems does not endow them with significantly more power than deterministic static proofs.

1.1.2 Argument Systems

Definition 1.2. An *argument system* for a language $\mathcal{L} \subseteq \{0,1\}^*$ is an interactive proof for $\mathcal{L}$, in which the soundness condition is only required to hold against prover strategies that run in polynomial time.

Argument systems were introduced by Brassard, Chaum, and Crépeau in 1986 [BCC88]. Unlike interactive proofs, argument systems are able to utilize cryptographic primitives. While a super-polynomial time prover may be able to break the primitive and thereby trick the verifier into accepting an incorrect answer, a polynomial time prover will be unable to break the primitive. The use of cryptography often allows argument systems to achieve additional desirable properties that are unattainable for interactive proofs, such as reusability (i.e., the ability for the verifier to reuse the same "secret state" to outsource many computations on the same input), zero-knowledge, public verifiability, etc. These properties will be discussed in more detail later in this survey.

1.2 Schwartz-Zippel Lemma

1.2.1 Terminolgy

For an m-variate polynomial g, the degree of a term of g is the sum of the exponents of the variables in the term. For example if $g(x_1,x_2) = 7x_1^2x_2 + 6x_2^4$, then the degree of the term $7x_1^2x_2$ is 3, and the degree of the term $6x_2^4$ is 4. The total degree of g is the maximum of the degree of any term of g.

1.2.2 The Lemma Itself

Interactive proofs frequently exploit the following basic property of polynomials, which is commonly known as the Schwartz-Zippel lemma [Sch80, Zip79].

Lemma 1.3 (Schwartz-Zippel Lemma)**.** *Let $\mathbb{F}$ be any field, and let $g : \mathbb{F}^m \to \mathbb{F}$ be a nonzero polynomial of total degree at most d. Then on any finite set $S \subseteq \mathbb{F}$,*

$$\Pr_{\mathbf{x} \leftarrow S^m}[g(\mathbf{x}) = 0] \leq d/|S|.$$

In words, if $\mathbf{x}$ is chosen uniformly at random from S^m, then the probability that $g(\mathbf{x}) = 0$ is at most $d/|S|$. In particular, any two distinct polynomials of total degree at most d can agree on at most $d/|S|$ fraction of points in S^m.

We will not prove the lemma above, but it is easy to find a proof online (see, e.g., the wikipedia article on the lemma). An easy implication of the Schwartz-Zippel lemma is that for any two distinct m-variate polynomials p and q of total degree at most d over $\mathbb{F}$, $p(\mathbf{x}) = q(\mathbf{x})$ for at most a $d/|\mathbb{F}|$ fraction of inputs. The lecture on Reed-Solomon fingerprinting exploited precisely this implication in the special case of univariate polynomials (i.e., $m = 1$).

1.3 Low Degree and Multilinear Extensions

Let $\mathbb{F}$ be any finite field, and let $f \colon \{0,1\}^v \to \mathbb{F}$ be any function mapping the v-dimensional Boolean hypercube to $\mathbb{F}$. A v-variate polynomial g over $\mathbb{F}$ is said to be an *extension* of f if g agrees with f at all Boolean-valued inputs, i.e., $g(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^v$.[1]

Preview: Why low-degree extensions are useful. One can think of a (low-degree) extension g of f as an error-corrected (or, at least, *distance-amplifying*) encoding of f in the following sense: if two Boolean functions f, f' disagree at even a single input, then any degree d extensions g, g' must differ *almost everywhere* (assuming $d \ll |\mathbb{F}|$). This is made precise by the Schwartz-Zippel lemma above, which guarantees that g and g' agree on at most $d/|\mathbb{F}|$ fraction of points in $\mathbb{F}^v$. This is entirely analogous to Reed-Solomon fingerprinting, which exploited the special case of Schwartz-Zippel for univariate polynomials. As we will see in the coming lectures, these distance-amplifying properties give the verifier surprising power over the prover. □

Definition 1.4. A multivariate polynomial g is *multilinear* if the degree of the polynomial in each variable is at most one.

[1] Later in this course, we will consider extensions of functions $f \colon \{0,\dots,M-1\}^v \to \mathbb{F}$ with $M > 1$. In this case, we say that $g \colon \mathbb{F}^v \to \mathbb{F}$ extends f if $g(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \{0,\dots,M-1\}^v$. Here, we interpret each number in $\{0,\dots,M-1\}$ as elements of $\mathbb{F}$ via any efficiently computable injection from $\{0,\dots,M-1\}$ to $\mathbb{F}$.

	0	1
0	1	2
1	1	4

Figure 1: A function f mapping $\{0,1\}^2$ to the field $\mathbb{F}_5$.

	0	1	2	3	4
0	1	2	3	4	5
1	1	4	2	0	3
2	1	1	1	1	1
3	1	3	0	2	4
4	1	0	4	3	2

Figure 2: The multilinear extension, $\widetilde{f}$ of f over $\mathbb{F}_5$.

For example, the polynomial $g(x_1,x_2) = x_1x_2 + 4x_1 + 3x_2$ is multilinear, but the polynomial $h(x_1,x_2) = x_2^2 + 4x_1 + 3x_2$ is not.

Throughout this course, we will frequently use the following fact.

Fact 1.5. *Any function $f\colon \{0,1\}^v \to \{0,1\}$ has a unique* multilinear *extension (MLE) over $\mathbb{F}$, and we reserve the notation $\widetilde{f}$ for this special extension of f.*

That is, $\widetilde{f}$, is the unique multilinear polynomial over $\mathbb{F}$ satisfying $\widetilde{f}(x) = f(x)$ for all $x \in \{0,1\}^v$. See Figure 2 for an example of a function and its multilinear extension.

Proof of Fact 1.5. Lemma 1.6 below demonstrates that for any function $f\colon \{0,1\}^v \to \mathbb{F}$, there is some multilinear polynomial that extends f. To show uniqueness of the multilinear extension of f, we want to show that if p and q are two multilinear polynomials such that $p(\mathbf{x}) = q(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^v$, then $p(\mathbf{x}) = q(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{F}^v$ (equivalently, we want to show that the polynomial $h := p - q$ is the identically 0 polynomial).

Observe that h is also multilinear, because it is the difference of two multilinear polynomials. Furthermore, the assumption that $p(\mathbf{x}) = q(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^v$ implies that $h(\mathbf{x}) = 0$ for all $\mathbf{x} \in \{0,1\}^v$. We now show that any such polynomial is identically 0.

So assume that h is a multilinear polynomial that vanishes on $\{0,1\}^v$. If h is not identically zero, then consider any term t in h of minimal degree. h must have at least one such term since h is not identically 0. For example, if $h(x_1,x_2,x_3) = x_1x_2x_3 + 2x_1x_2$, then the term $2x_1x_2$ is of minimal degree, since it has degree 2, and h has no terms of degree 1 or 0.

Now consider the input z obtained by setting all of the variables in t to 1, and all other variables to 0 (in the example above, $z = (1,1,0)$). At input z, term t is non-zero because all of the variables appearing in term t are set to 1. For instance, in the example above, the term $2x_1x_2$ evaluates to 2 at input $(1,1,0)$).

Meanwhile, all other terms contain of h at least one variable that is not in term t (otherwise, t would not be of minimal degree in h). Since z sets all variables not in t to 0, this means that all terms in h other that t evaluate to 0 at z. It follows that $h(z) \neq 0$ (e.g., in the example above, $h(z) = 2$).

This contradicts the assumption that $h(\mathbf{x}) = 0$ for all $\mathbf{x} \in \{0,1\}^v$. We conclude that any multilinear polynomial h that vanishes on $\{0,1\}^v$ must be identically zero, as desired. □

MLEs have a particularly simple representation, given by Lagrange interpolation:

Lemma 1.6. *Let $f\colon \{0,1\}^v \to \mathbb{F}$ be any function. Then, as formal polynomials,*

$$\widetilde{f}(x_1,\ldots,x_v) = \sum_{\mathbf{w}\in\{0,1\}^v} f(\mathbf{w}) \cdot \chi_{\mathbf{w}}(x_1,\ldots,x_v), \tag{1}$$

where, for any $\mathbf{w} = (w_1, \ldots, w_v)$*,*

$$\chi_{\mathbf{w}}(x_1, \ldots, x_v) := \prod_{i=1}^{v} (x_i w_i + (1 - x_i)(1 - w_i)). \tag{2}$$

Proof. For any vector $\mathbf{w} \in \{0,1\}^v$, $\chi_{\mathbf{w}}$ is the unique multilinear polynomial satisfying: $\chi_{\mathbf{w}}(\mathbf{w}) = 0$, and $\chi_{\mathbf{w}}(\mathbf{y}) = 0$ for all other vectors $\mathbf{y} \in \{0,1\}^v$. Clearly the right hand side of Equation (1) is a multilinear polynomial in $(x_1, \ldots, x_v)$, so in order to show that it is equal to the unique MLE of f, we need only show that $\sum_{\mathbf{w} \in \{0,1\}^v} f(\mathbf{w}) \cdot \chi_{\mathbf{w}}(\mathbf{y}) = f(\mathbf{y})$ for all *Boolean* vectors $\mathbf{y} \in \{0,1\}^v$; this is immediate from the previous sentence. □

Suppose that the verifier is given as input the values $f(\mathbf{w})$ for all $n = 2^v$ Boolean vectors $\mathbf{w} \in \{0,1\}^v$. Equation (1) yields two efficient methods for evaluating $\tilde{f}$ at any point $\mathbf{r} \in \mathbb{F}^v$, The first method was described in [CTY10]: it requires $O(n \log n)$ time, and allows $\mathcal{V}$ to make a single streaming pass over the $f(\mathbf{w})$ values while using $v + 1 = O(\log n)$ words of space. The second method is due to Vu et al. [VSBW13]: it shaves a logarithmic factor off of $\mathcal{V}$'s runtime, bringing it down to linear time, i.e., $O(n)$, but increases $\mathcal{V}$'s space usage to $O(n)$.

Lemma 1.7 ([CTY10])**.** *Fix a positive integer* v *and let* $n = 2^v$*. Given as input* $f(\mathbf{w})$ *for all* $\mathbf{w} \in \{0,1\}^v$ *and a vector* $\mathbf{r} \in \mathbb{F}^{\log n}$*,* $\mathcal{V}$ *can compute* $\tilde{f}(\mathbf{r})$ *in* $O(n \log n)$ *time and* $O(\log n)$ *words of space with a single streaming pass over the input (regardless of the order in which the* $f(\mathbf{w})$ *value are presented).*

Proof. $\mathcal{V}$ can compute the right hand side of Equation (1) incrementally from the stream by initializing $\tilde{f}(\mathbf{r}) \leftarrow 0$, and processing each update $(\mathbf{w}, f(\mathbf{w}))$ via:

$$\tilde{f}(\mathbf{r}) \leftarrow \tilde{f}(\mathbf{r}) + f(\mathbf{w}) \cdot \chi_{\mathbf{w}}(\mathbf{r}).$$

$\mathcal{V}$ only needs to store $\tilde{f}(\mathbf{r})$ and $\mathbf{r}$, which requires $O(\log n)$ words of memory (one for each entry of $\mathbf{r}$). Moreover, for any $\mathbf{w}$, $\chi_{\mathbf{w}}(\mathbf{r})$ can be computed in $O(\log n)$ field operations (see Equation (2)), and thus $\mathcal{V}$ can compute $\tilde{f}(\mathbf{r})$ with one pass over the stream, using $O(\log n)$ words of space and $O(\log n)$ field operations per update. □

The algorithm of Lemma 1.7 computes $\tilde{f}(r)$ by evaluating each term on the right hand side of Equation (1) independently in $O(v)$ time and summing the results. This results in a total runtime of $O(v \cdot 2^v)$. The following lemma gives an even faster algorithm, running in time $O(2^v)$. It's speedup relative to Lemma 1.7 is obtained by *not* treating each term of the sum independently. Rather, using dynamic programming, Lemma 1.8 computes $\chi_{\mathbf{w}}(\mathbf{r})$ *for all* 2^v vectors $\mathbf{w} \in \{0,1\}^v$ in time $O(2^v)$.

Lemma 1.8 ([VSBW13])**.** *Fix a postive integer* v*, and let* $n = 2^v$*. Given as input* $f(\mathbf{w})$ *for all* $\mathbf{w} \in \{0,1\}^v$ *and a vector* $\mathbf{r} = (r_1, \ldots, r_v) \in \mathbb{F}^{\log n}$*,* $\mathcal{V}$ *can compute* $\tilde{f}(\mathbf{r})$ *in* $O(n)$ *time and* $O(n)$ *space.*

Proof. Notice the right hand side of Equation (1) expresses $\tilde{f}(\mathbf{r})$ as the inner product of two n-dimensional vectors, where the $\mathbf{w}$'th entry of the first vector is $f(\mathbf{w})$ and the $\mathbf{w}$th entry of the second vector is $\chi_{\mathbf{w}}(\mathbf{r})$. This inner product can be computed in $O(n)$ time given a table of size n whose $\mathbf{w}$th entry contains the quantity $\chi_{\mathbf{w}}(\mathbf{r})$. Vu et al. show how to build such a table in time $O(n)$ using memoization.

The memoization procedure consists of $v = \log n$ stages, where Stage j constructs a table $A^{(j)}$ of size 2^j, such that for any $(w_1, \ldots, w_j) \in \{0,1\}^j$, $A^{(j)}[(w_1, \ldots, w_j)] = \prod_{i=1}^{j} \chi_{w_i}(r_i)$. Notice $A^{(j)}[(w_1, \ldots, w_j)] = A^{(j-1)}[(w_1, \ldots, w_{j-1})] \cdot (w_j r_j + (1 - w_j)(1 - r_j))$, and so the jth stage of the memoization procedure requires time $O(2^j)$. The total time across all $\log n$ stages is therefore $O(\sum_{j=1}^{\log n} 2^j) = O(2^{\log n}) = O(n)$. □

References

[Bab85] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *STOC*, pages 421–429. ACM, 1985.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[CTY10] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:159, 2010.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *STOC*, pages 59–68. ACM, 1986.

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

[VSBW13] Victor Vu, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 223–237. IEEE Computer Society, 2013.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.