Technion IIT
Department of Computer Science
Fall 2013-14

Course 236646:

Succinct Computational Integrity (SCI) and Probabilistically Checkable Proofs (PCP)

Eli Ben-Sasson

Table of Contents

Lecture 7: Soundness boosting via gap amplification

References

Lecture 1

Introduction

October 15 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

This lecture is devoted to explaining the two versions of the PCP theorem that we will study in this course. We start next with informal description and motivation, followed by formal definitions of the theorems that address these goals.

1.1 Succinct Computational Integrity and Hardness of Approximation

The introduction of interactive proofs Babai and Moran [1988]; Goldwasser et al. [1989] led quickly to many results that studied various forms of proof systems. This course focuses on Probabilistically Checkable Proofs (PCPs), also known as holographic, or transparent, proofs. These are proof systems in which a prover writes a proof for a given statement and the verifier then "tests", or "verifies" the correctness of the proof via an efficient and randomized procedure that, in particular, reads only a negligible part of the proof and yet is "valid", meaning that correct proofs of true statements are accepted with high probability and any "proof" of a false statement is rejected with high probability. Early on, two distinct possible use cases were offered for such proofs:

Succinct Computational Integrity Babai et al. [1991] pointed out that such proof systems can offer a succinct solution to the computational integrity problem. We start with a Verifier V holding the description of a nondeterministic machine M. She wishes to have an untrusted prover P simulate the execution of M for T steps and report to her the outcome of this computation. The problem of computational integrity is that if P cannot be trusted, how can V trust his output? A trivial solution is for V to ask P to describe his nondeterministic choices and then simulate M's execution. Babai et al. [1991] showed that, using PCPs (which they called holographic proofs there), V can obtain reasonable confidence in the integrity of P's computation by running a (randomized) verification protocol that costs only $\text{poly}(|M|, \log T)$ steps, given random access to bits of the proof. Even better, P can generate the relevant proof in time $T \cdot \text{poly}(|M|, T)$.

Hardness of Approximation Many NP-complete problems start with a set of constraints over n variables and ask whether there exists an assignment that satisfies all of the constraints. Examples of such problems are graph-coloring, 3SAT and SetCover. In lieu of polynomial-time algorithms for solving these problems, a natural question to ask is whether there exist polynomial-time algorithms that are guaranteed to find an assignment that sat-

isfies a non-negligible fraction of the constraints, where this fraction is measured relative to the (possibly not efficiently attainable) optimal assignment, one that maximizes the fraction of satisfied constraints. Feige et al. [1996] showed that the PCP Theorem implies that for certain optimization problems, no nontrivial solutions exists unless $\mathbf{P} = \mathbf{NP}$.

1.2 Complexity Classes defined by PCP verifiers

At the core of a PCP system lies a *verifier* — the randomized machine that checks the perceived correctness of proofs. Our intuitive definition of a "proof" is of a sequence of ℓ symbols from some finite alphabet Σ. However, since we will severely limit the number of symbols read from a proof, we prefer to view it as an *oracle*, i.e., as a function $\pi : \{1, \ldots, \ell\} \to \Sigma$. This way, a machine with oracle access to π can specify an index i of some entry of the proof and obtain its value within $O(1)$ steps, without needing to move its reading head to position i on the tape.

Definition 1.1 (PCP-Verifier)**.** A *PCP-verifier*, or simply, *verifier*, is a randomized Turing machine V with access to *proof oracle*, or, simply *proof* π. On input x and random coin tosses $R \in \{0,1\}^*$, V makes a number of queries to π and outputs either accept or reject. We denote by $V^\pi[x;R]$ the output of V on input x, proof π and random coins R. If the queries and decision predicate do not depend on answers to previous queries we say V is *nonadaptive*. (Most verifiers we shall encounter here are nonadaptive.)

Being interested in efficient verifiers, we are going to limit some of their computational resources such as the running time, and the length and number of bits read from the proof. Additionally, we will require that the verifier make a correct decision with sufficient probability. Good proofs of correct statements must be accepted with a minimal probability called the *completeness* parameter. Purported proofs of incorrect statements must be rejected with a minimal probability known as the *soundness* parameter. The probability of error in both the completeness and soundness cases depend on the random coin tosses of the verifier. The restrictions and the completeness and soundness parameters may depend on the length of the input statement. Once completeness and soundness, and the limitations on computational resources, are fixed, we have effectively defined a complexity class. A language belongs to it if there exists a verifier that operates under the computational restrictions and nevertheless succeeds in judging correctly the validity of proofs with sufficiently high probability.

Definition 1.2 (PCP-defined class of languages)**.** Given a list of computational restrictions, a completeness function $c : \mathbb{N}^+ \to [0,1]$ and a soundness function $s : \mathbb{N}^+ \to [0,1]$, the complexity class $\mathbf{PCP}\left(\text{list of restrictions} \;\middle|\; \begin{matrix} c(n) \\ s(n) \end{matrix} \right)$ is the set of languages $L \subseteq \Sigma^*$ for which there exists a verifier V_L satisfying for every integer n and $x \in \Sigma^n$:

- **Operation:** V_L on input x does not violate the listed restrictions.

- **Completeness:** If $x \in L$ there exists a proof π such that

$$\Pr_R[V_L^\pi[x;R] = \mathsf{accept}] \geq c(n).$$

- **Soundness:** If $x \notin L$ then for every proof π,

$$\Pr_R[V_L^\pi[x;R] = \mathsf{reject}] \geq s(n).$$

1.3 Two variants of the PCP Theorem

The following variant of the PCP Theorem gives nearly optimal proof length and prover and verifier running time. Thus, it is more tailored for positive applications to efficient generation and checking of proofs and computations and to succinct computational integrity.

Theorem 1.3 (PCP Theorem — short proofs)**.** *There exists an absolute constant $\epsilon > 0$ such that for every proper complexity function $T : \mathbb{N}^+ \to \mathbb{N}^+$,*

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} \ell(n) & \leq & T(n)\cdot \mathrm{polylog}(T(n)) & & & \\ t(n) & \leq & \mathrm{polylog}(T(n)) & & & \\ r(n) & = & \log(\ell(n)) + O(1) & c & = & 1 \\ q & \leq & 2 & s & \geq & \epsilon \\ \Sigma & = & \{0,1,2\} & & & \\ \mathsf{nonadaptive} & & & & & \end{array}\right).$$

Where

- *$\ell(n)$ is the length of the proof, or, formally, the largest index of a proof-symbol that may be queried by V when given input of length n.*
- *$t(n)$ denotes the running time of V as a function of the input length.*
- *$r(n)$ is the number of random bits required by V on input of length n.*
- *q denotes the number of queries V makes to the proof oracle.*
- *Σ denotes the alphabet of the proof. Each query is answered with a single element from this alphabet.*
- nonadaptive *means that the set of queries made to the proof and the decision process based on the answers given by the oracle depend only on x and the random coins R, and not on answers given by the oracle to previous queries.*

Furthermore, for any $L \in \mathbf{NTIME}\left(T(n)\right)$, there exists a deterministic Prover machine P_L running in time $T(n)\mathrm{polylog}(T(n))$ that on input $x \in L$ and nondeterministic witness w for x, produces a proof $\pi = \pi_{x,w}$ that is accepted by V_L with probability 1.

Next we give the version suited for applications regarding hardness of approximation.

Theorem 1.4 (PCP Theorem — query efficient and sound)**.** *For every proper complexity function $T : \mathbb{N}^+ \to \mathbb{N}^+$ and any $\epsilon > 0$,*

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} q & \leq & 3 & & & \\ \Sigma & = & \{0,1\} & & & \\ \mathsf{nonadaptive} & & & & & \\ \mathsf{query-type} & & XOR & c & \geq & 1-\epsilon \\ t(n) & \leq & \mathrm{poly}(T(n)) & s & \geq & \frac{1}{2}-\epsilon \\ \ell(n) & \leq & T(n)^{1+o(1)} & & & \\ r(n) & \leq & \log(\ell(n)) + O(1) & & & \end{array}\right).$$

The notation for the list of restrictions is the same as in Theorem 1.3, and $\mathsf{query-type}$ *denotes the class of computations performed by the verifier after receiving the query answers. In the case of XOR, the computation depends only on the XOR of the (three) answer bits.*

A few remarks about the previous theorem are due. Notice that improving the completeness or soundness seems unlikely (assuming $\mathbf{P} \neq \mathbf{NP}$). If $c = 1$ and all other parameters are left unchanged then $\mathbf{P} = \mathbf{NP}$ because deciding whether a sequence of bits satisfies a collection of XOR constraints is equivalent to solving a system of linear equations over the two-element field and can be done (say, by Gaussian elimination) in polynomial time. Similarly, if s must be less than $1/2$ because a random proof (where each bit is selected by a random coin toss) will be accepted by V with probability $1/2$. The optimality of the soundness and completeness in conjunction with the small query complexity and simplicity of the query type have far reaching implications to our understanding the limitations of approximation algorithms and this will be discussed later.

1.4 Bibliographical notes

The story of the PCP Theorem and the way its proof evolved is quite interesting. An illustrated and entertaining description of this history can be found in O'Donnell [Autumn 2005]. Each of the two variants of the PCP Theorem stated in this lecture rely on several important works. The basic statement of a constant-query PCP characterization of **NP** appeared in Arora et al. [1998] and relies on Arora and Safra [1998]. The application of the PCP Theorem to efficient program checking (to be discussed in the following lecture) appeared first in Babai et al. [1991] and the implication of the PCP theorem to hardness of approximation was first observed in Feige et al. [1996]. The length-efficient PCP variant presented in Theorem 1.3 together with its efficient prover is a combination of Ben-Sasson and Sudan [2005]; Dinur [2007]; Ben-Sasson et al. [2005a]; Mie [2009]; Ben-Sasson et al. [2013b,a]. The query efficient PCP variant presented in Theorem 1.4 appeared is from Håstad [1997]; Raz [1998]; Bellare et al. [1998]; Moshkovitz and Raz [2010].

Lecture 2

Local codes

October 22 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

The way a verifier checks in time t the integrity of a computation run by the prover for $T \gg t$ steps, goes by asking the prover to *(i)* encode all relevant information about the computation via a special error correcting code, and *(ii)* reducing the problem of computational integrity to the simpler problem of codeword integrity. We start by describing the kind of codes that are amenable to "local" or "succinct" integrity testing, as needed for part *(i)*, and later will deal with the reduction described in *(ii)*.

2.1 Error correcting codes

We start with the definition of an *error correcting code*, as introduced in the 1940's in the pioneering work of Golay, Hamming and Shannon (cf. MacWilliams and Sloane [1977]). Since we plan to "test" codewords at random locations we prefer to think of a codeword as a function from indices to symbols.

Definition 2.1 (Error-Correcting Code)**.** Let D, Σ be finite sets and $C = \{f : D \to \Sigma\}$ a set of functions from the domain D to the alphabet Σ. We say C is an $(n, k, d)_\Sigma$-*error correcting code* when

- The *blocklength* of C is $n = |D|$.
- The size of C is $|\Sigma|^k$. We call k the *information length* and k/n is the *rate* of C.
- Every $f, g \in C$, $f \neq g$, have (Hamming) distance at least d, where the (Hamming) distance of f and g is

$$\mathsf{Hamm}(f, g) = |\{x \in D | f(x) \neq g(x)\}|$$

  and the *relative (Hamming) distance* is $\mathsf{rHamm}(f, g) = \mathsf{Hamm}(f, g)/n$.

When Σ is a finite field $\mathbb{F}$ and C is a linear space over $\mathbb{F}$ we call it an $[n, k, d]_\mathbb{F}$-linear error correcting code (notice the slightly different bracket notation). Recall that C is linear over $\mathbb{F}$ if for every $f, g \in C$ and $\alpha, \beta \in \mathbb{F}$ the function $\alpha f + \beta g$ belongs to C, where $(\alpha f + \beta g)(x) = \alpha f(x) + \beta g(x)$.

2.2 Testers for codeword integrity

We reduce the problem of computational integrity to that of codeword integrity via use of a *codeword integrity tester*, or, simply, *tester*. We limit our discussion to *nonadaptive* testers,

meaning their queries and decision predicate do not depend on answers given to previous queries. All testers we will encounter here are nonadaptive and henceforth we will drop further mention of this property.

Definition 2.2 (Tester)**.** Let $C = \{f : D \to \Sigma\}$ be a code with blocklength $n = |D|$ over alphabet Σ. A (nonadaptive) (t, r, q)-*restricted tester* for C is a randomized algorithm T that receives a randomness input $R \in \{0,1\}^r$ and, after t steps, outputs a sequence of q queries $x_1, \ldots, x_q \in D$ along with a *decision predicate* which is a mapping $\mathsf{Dec} : D^q \times \Sigma^q \to \{\mathsf{accept}, \mathsf{reject}\}$.
We say T has *completeness* $c \in [0,1]$ and *soundness function* $s : [0,1] \to [0,1]$ for C if the following conditions hold:

- **Completeness:** $f \in C$ implies

$$\Pr_{R \in \{0,1\}^r} [\mathsf{Dec}\,(x_1, \ldots, x_q, f(x_1), \ldots, f(x_q)) = \mathsf{accept}] \geq c.$$

  Notice that $x_1, \ldots, x_q$ and Dec may depend on R.

- **Soundness:** $f \notin C$ implies

$$\Pr_{R \in \{0,1\}^r} [\mathsf{Dec}\,(x_1, \ldots, x_q, f(x_1), \ldots, f(x_q)) = \mathsf{reject}] \geq s(\mathsf{rHamm}(f, C)),$$

  where $\mathsf{rHamm}(f, C) = \min\{\mathsf{rHamm}(f, g) | g \in C\}$ is the minimal relative distance of f from all codewords of C.

Finally, we say that C is a (q, ϵ, δ)-locally testable code (LTC) if it has a q-query tester with completeness $c = 1$ and soundness function satisfying $s(\delta') \geq \epsilon$ for all $\delta' > \delta$.

Remark (Linear codes require only linear testers). For a linear code C over a field $\mathbb{F}$, let $C^\perp$ be the space dual to C,

$$C^\perp = \{g : D \to \mathbb{F} | \sum_{x \in D} f(x) \cdot g(x) = 0 \text{ for all } f \in C\}$$

where arithmetic operations are over $\mathbb{F}$. A *linear tester* for C is defined by a distribution μ over $C^\perp$. To use such a tester, sample g from $C^\perp$ according to μ and accept f iff $\sum_{x \in D} f(x) \cdot g(x) = 0$. By definition of $C^\perp$ this test has completeness 1, and, for instance, if μ is the uniform distribution over $C^\perp$ then $f \notin C$ is rejected with probability $1 - 1/|\mathbb{F}|$ (This is because the subset of $C^\perp$ that is orthogonal to f is a subspace of $C^\perp$ of co-dimension 1.)
Ben-Sasson et al. [2005b] showed that if C has a tester with query complexity q, completeness c, and soundness s, then it has a linear tester with query complexity q, perfect completeness 1, and soundness $s + (1 - c)$.

Problem 2.1. Let C be a linear code that is a (q, ϵ, δ)-LTC for some $\epsilon > 0$ and $\delta = \mathsf{rHamm}(C)/3$, where $\mathsf{rHamm}(C)$ is the minimal relative distance of C. Prove: The minimal distance of $C^\perp$ is at most q.

2.3 The BLR Theorem: Linearity functions are locally testable

Our first example of an LTC is due to Blum et al. [1990] and says that if $C = \{\phi : G \to H\}$ is the code whose codewords are homomorphisms from a group G to a group H, then C is a $(3, \epsilon, \delta)$-LTC for fixed positive ϵ, δ that are independent of G and H. For concreteness we deal only with the case of $G = (\mathbb{F}_2^k, +)$ and $H = (\mathbb{F}_2, +)$ where $\mathbb{F}_2$ is the two-element field, the relevant code is called the *Hadamard code* of dimension k, denoted here by Had_k. The proof generalizes to any G and H.
We use the notation $\langle f, g\rangle_D$ for $\sum_{x\in D} f(x)g(x)$, when D is clear from context we simply write $\langle f, g\rangle$. We also use $[k]$ for $\{1, \ldots, k\}$.

Problem 2.2. Prove: A function $f : \mathbb{F}_2^k \to \mathbb{F}_2$ is a homomorphism iff there exists $a \in \mathbb{F}_2^k$ such that $f(x) = \langle a, x\rangle_{[k]} = \sum_{i=1}^k a_i x_i$ for all $x \in \mathbb{F}_2^k$.

Problem 2.3. Prove: the minimal distance of the k-dimensional Hadamard code is 2^{k-1}. (Extra: What is the minimal distance of $C = \{\phi : G \to H\}$, as a function of $|G|, |H|$?)

The following tester for the Hadamard code was suggested by Blum et al. [1990].

Definition 2.3 (BLR-Tester)**.** Let T be the tester which, on input $(x, y) \in \mathbb{F}_2^{2k}$ outputs the queries $(x, y, x + y)$ and accepts the answers $f(x), f(y), f(x + y)$ iff $f(x) + f(y) = f(x + y)$.

Problem 2.4. Formulate and prove: The BLR-tester has (perfect) completeness 1.

Next we prove the (harder) bound on the soundness function.

Theorem 2.4. *If*

$$\Pr_{x,y\in F_2^k}[f(x) + f(y) \neq f(x + y)] = \varepsilon < 2/9, \tag{1}$$

Then $\mathsf{rHamm}(f, \mathsf{Had}_k) \leq 2\varepsilon$.

Problem 2.5. Use the previous theorem to bound from below the soundness function of the BLR-Tester T.

2.4 Proving the BLR Theorem

For the rest of this session we assume Equation 1 and present a word maj_f that is *(i)* 2ε-close to f and *(ii)* belongs to Had_k. The first part is relatively easy to prove and the second part is somewhat harder.

Definition 2.5 (Majority decoding)**.** Given $f : \mathbb{F}_2^k \to \mathbb{F}_2$ let $\mathsf{maj}_f : \mathbb{F}_2^k \to \mathbb{F}_2$ be defined by setting the x^{th} entry of maj_f to be the most common value of $f(x + y) + f(y)$. Formally,

$$\mathsf{maj}_f(x) = \mathsf{majority}\{f(x + y) + f(y) | y \in F_2^k\}.$$

Problem 2.6. Prove part *(i)* above: $\mathsf{rHamm}(w, w') \leq 2 \cdot \Pr_{x,y}[f(x) + f(y) \neq f(x + y)]$.

For the rest of this session we'll focus on part *(ii)* and prove $\mathsf{maj}_f \in \mathsf{Had}_k$. Our first step is to show that every entry $\mathsf{maj}_f(x)$ is selected by an overwhelming majority.

Problem 2.7. Prove: For all $x \in F_2^k$,

$$\Pr_y \left[f(x+y) + f(y) = \mathsf{maj}_f(x) \right] > 2/3.$$

Hints:

- Consider $y, z \in F_2^k$ selected randomly and independently.
- Show $\Pr_{y,z} [f(x+y) + f(z) = f(x+z) + f(y)] = \Pr_{y,z} [f(x+y) + f(y) = f(x+z) + f(z)] > 5/9$.
- Notice $\Pr_{y,z} [f(x+y) + f(y) = f(x+z) + f(z)]$ is the probability that two independent random "votes" for $\mathsf{maj}_f(x)$ agree. The probability that two independent identically distributed random variables agree on their value is the *collision probability* of the distribution and is useful for various probabilistic arguments, as we'll see next.
- Prove: If a $\{0,1\}$-valued random variable Y satisfying (w.l.o.g.) $\Pr[Y = 1] \geq 1/2$ has collision probability greater than $5/9$, then $\Pr[Y = 1] > 2/3$.

Problem 2.8. Using the previous problem prove that for any $x, y \in F_2^k$ we have $\mathsf{maj}_f(x) + \mathsf{maj}_f(y) = \mathsf{maj}_f(x+y)$. Conclude $\mathsf{maj}_f \in \mathsf{Had}$.

Lecture 3

Hadamard code based PCP of exponential length

October 28 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

We now proceed to use the local testability of the Hadamard code to prove the following version of the PCP Theorem, one that has constant query complexity and soundness, but very long proofs.

Theorem 3.1. *There exists a constant $\rho > 0$ such that for every proper complexity function $T : \mathbb{N}^+ \to \mathbb{N}^+$,*

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} \ell(n) & \leq & 2^{\tilde{O}(T(n))^2)} & & & \\ t(n) & \leq & \tilde{O}(T(n))^2) & & & \\ r(n) & = & \tilde{O}(T(n))^2) & c & = & 1 \\ q & = & 16 & s & \geq & \rho \\ \Sigma & = & \{0,1\} & & & \\ \text{nonadaptive} & & & & & \end{array}\right).$$

The notation for the list of restrictions is the same as in Theorem 1.3, and $\tilde{O}(N)$ means $O(N \cdot \log^{O(1)} N)$.

As explained in the first lecture, we shall reduce the problem of checking satisfiability of a boolean circuit to problems about error correcting codes. Specifically, we shall use the local testability of the Hadamard code, along with other "local" properties of this code that are developed next.

3.1 Locally decodable codes

We begin with a definition.

Definition 3.2 (Local decoding)**.** Let $C = \{f : D \to \mathbb{F}\}$ be an $[n, k, d]$-linear error correcting code over a field $\mathbb{F}$, and let $G \in \mathbb{F}^{n \times k}$ be its *generating matrix*, meaning that $C = \{G \cdot x | x \in \mathbb{F}_2^k\}$. We thus think of G as the transformation that maps a k-symbol long message m to a unique codeword $f_m : D \to \mathbb{F}$. Let $g : \mathbb{F}^k \to \mathbb{F}$ be a function defined on messages. We say g can be *locally decoded* from C with query complexity q, soundness δ and error ϵ — shortly, a (q, δ, ϵ)-LDC for g, if there exists a randomized process P that makes at most q queries into a purported codeword $f : D \to \mathbb{F}$ and outputs a symbol $\sigma \in \mathbb{F}$, and for which the following holds. If f is δ-close to a codeword f_m,

$$\Pr\left[\sigma \neq g(m)\right] \leq \epsilon.$$

The probability above is over the random choices of P. We similarly say C is a (q, δ, ϵ)-LDC for a family of functions if it is a (q, δ, ϵ)-LDC for each function in the family.

Problem 3.1. Prove: For any $\delta \in (0, 1/4)$ there exists ϵ such that the Hadamard code is a (q, δ, ϵ)-LDC for the family of $\mathbb{F}_2$-linear functions, where q is a fixed constant. What are ϵ and q?

The following problem will be pivotal in the arithmetization, i.e., in reducing questions about computational integrity to questions about local testability and decodability of the Hadamard code.

Problem 3.2. Let $f : \mathbb{F}_2^{k \times k} \to \mathbb{F}_2$ be the Hadamard encoding of a message $m \in \mathbb{F}_2^{k \times k}$ where we think of m as a $k \times k$ matrix. Prove :There exists a randomized proceduce that makes a constant number of queries to f and satisfies these properties:

- If $m = a \cdot a^\top$ for some $a \in \mathbb{F}_2^k$ then the procedure accepts with probability 1.
- If $m \neq a \cdot a^\top$ for some $a \in \mathbb{F}_2^k$ then the procedure rejects with probability $\geq 1/4$.

Recall $m = a \cdot a^\top$ means that the (i, j)-entry of m, denoted $m[i, j]$, equals $a_i \cdot a_j$. In other words, m is a rank-1 matrix formed by taking the outer-product of a with itself. Notice that a being $\{0, 1\}$-valued implies $\mathsf{diag}(m) = a$ where $\mathsf{diag}(m) = (m[1, 1], m[2, 2], \ldots, m[k, k])$. Hints:

- Argue that the following equation holds for all $r, s \in \mathbb{F}_2^k$ if and only if $m = a \cdot a^\top$ for some $a \in \mathbb{F}_2^k$:
$$\langle r, \mathsf{diag}(m) \rangle \cdot \langle \mathsf{diag}(m), s \rangle = \langle r \cdot s^\top, m \rangle$$
where $\langle r \cdot s^\top, m \rangle = \sum_{i,j=1}^k (r \cdot s^\top)[i, j] \cdot m[i, j]$.
- What is the probability that equality holds in the above equation when m is not $a \cdot a^\top$ and r, s are picked uniformly at random from $\mathbb{F}_2^k$?

3.2 Arithmetization

The starting point of our reduction is an instance of circuit-SAT:

Definition 3.3. A boolean circuit ϕ with k gates $\phi_1, \ldots, \phi_k$, is given by a directed acyclic graph with k vertices, fan-in 2, and vertices are labeled by a gate-type which can be AND or NOT. An assignment is a mapping $\alpha : [k] \to \{0, 1\}$ and we say α satisfies ϕ iff

- *ϕ_i is the AND of $\phi_j, \phi_{j'}$:* $\alpha(i) = \alpha(j) \cdot \alpha(j')$.
- *ϕ_i is the NOT of ϕ_j:* $\alpha(i) = 1 - \alpha(j)$
- *Output:* $\alpha(k) = 1$

We say ϕ is satisfiable if it has a satisfying assignment and otherwise we say ϕ is unsatisfiable. Finally, CIRCUIT-SAT is the language containing all satisfiable circuits.

To "arithmetize" this set of constraints, i.e., to be able to use it in a Hadamard-based PCP construction, we expand the information in α quadratically, and seek an assignment to a larger problem that is nevertheless equivalent to the former one. This latter problem will be equivalent to Problem 3.2

Definition 3.4. $\beta \in \mathbb{F}_2^{k \times k}$ satisfies ϕ if the following holds :

- $\forall i, j : \beta_{i,j} = \beta_{i,i} \cdot \beta_{j,j}$
- *ϕ_i is an AND constraint:* $\beta_{i,i} = \beta_{j,k}$
- *ϕ_i is a NOT constraint:* $\beta_{i,i} = \beta_{j,j} + 1$
- *Output:* $\beta_{k,k} = 1$

Problem 3.3. Prove: $\alpha \in \mathbb{F}_2^k$ satisfies ϕ according to Definition 3.3 $\iff \beta = \alpha \cdot \alpha^T$ and β satisfies ϕ according to Definition 3.4.

Problem 3.4. Prove Theorem 3.1, using Theorem 2.4 and the previous problems in this session. Further hints:

- You may quote the following Theorem: Let $L \in \mathbf{NTIME}(T(n))$. Then there exists a deterministic reduction running in time $\tilde{O}(T(n))$ from L to CIRCUIT-SAT. In particular, an instance of L of size n is reduced to an instance of CIRCUIT-SAT of size $\tilde{O}(T(n))$.
- Use Problem 3.3 to reduce circuit to instance of Definition 3.4
- Ask prover to provide Hadamard encoding of matrix via a codeword f.
- Use Theorem 2.4 to test the proof
- Assuming linearity-test past, assume f is close to a Hadamard codeword.
- Use the problems of this session to test all constraints of Definition 3.4 via a constant number of queries to f.

Lecture 4

Low degree testing of multivariate polynomials

November 12 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

Previously we obtained a PCP system based on the Hadammard code, its verifier makes a constant number of queries but the proof is exponentially long.
In the next few sessions we will study algebraic tools that reduce proof length while maintaining a small (and even constant) number of queries.

4.1 Reed-Solomon (RS) and Reed-Muller (RM) codes

In what follows, a *monomial* M is an expression of the form $M \triangleq \prod_{i=1}^{m} X_i^{d_i}$. It's *degree in the i'th variable* is $\deg_{X_i}(M) \triangleq d_i$, it's *individual degree* is $\deg_{\mathbf{i}}(M) \triangleq \max_i d_i$, and its *total degree*, or, simply, *degree* is $\sum_i d_i$. A polynomial P is a sum of monomials $P = \sum_j M_j$, and it's (ith/individual/total) degree is the maximal degree of a monomial M_j appearing in it.

Definition 4.1 (RM and RS codes)**.** Let $\mathbb{F}$ be a finite field and m, d be integers. The m-variate, degree-d Reed-Muller (RM) code over $\mathbb{F}$ is

$$\mathsf{RM}[\mathbb{F}, d, m] = \{f : \mathbb{F}^m \to \mathbb{F} | \deg(f) \leq d\}$$

In other words, f belongs to $\mathsf{RM}[\mathbb{F}, d, m]$ iff there exists a polynomial $P \in \mathbb{F}[X_1, \ldots, X_m]$ of degree at most d such that $P(x_1, \ldots, x_m) = f(x_1, \ldots, x_m)$.
When $m = 1$ the code is called a *Reed-Solomon (RS)* code and denoted $\mathsf{RS}[\mathbb{F}, d]$.

Problem 4.1. Prove the so-called "Schwartz-Zippel Lemma", which is the following lower bound on the distance of $\mathsf{RM}[\mathbb{F}, d, m]$: If $f \in \mathsf{RM}[\mathbb{F}, d, m]$ is non-zero, then it's relative hamming weight is at least $1 - d/|\mathbb{F}|$. Hint: Induction on m.

4.2 Low degree testing

There is a natural test for RM-codes, suggested by the following problem. To state it we need some notation. A *line* in $\mathbb{F}^m$ is a function $\mathsf{line}_{a,b} : \mathbb{F} \to \mathbb{F}^m$ given by $\mathsf{line}_{a,b}(x) = a \cdot x + b$ where $a = (a_1, \ldots, a_m) \in \mathbb{F}^m$ is the *slope* and $b = (b_1, \ldots, b_m)$ is the *shift*. Similarly, a *plane* in $\mathbb{F}^m$ is a function $\mathsf{plane}_{a,a',b} : \mathbb{F}^2 \to \mathbb{F}$ given by $\mathsf{plane}_{a,a',b}(x, y) = a \cdot x + a' \cdot y + b$. (Higher-dimensional planes are similarly defined). The *restriction* of a m-variate function $f : \mathbb{F}^m \to \mathbb{F}$ to a line $\mathsf{line}_{a,b}$, or a plane $\mathsf{plane}_{a,a'b}$ respectively, is

$$f|_{\mathsf{line}_{a,b}}(x) = f(\mathsf{line}_{a,b}(x)), \quad f|_{\mathsf{plane}_{a,b}}(x, y) = f(\mathsf{plane}_{a,a',b}(x, y)), \quad \text{respectively.}$$

Problem 4.2. Assume $dm < |\mathbb{F}|$. Then $f \in \mathsf{RM}[\mathbb{F}, d, m]$ if and only if for every $a, b \in \mathbb{F}^m$ we have $f|_{\mathsf{line}_{a,b}} \in \mathsf{RS}[\mathbb{F}, d]$, which happens iff for every $a, a', b \in \mathbb{F}^m$ we have $f|_{\mathsf{plane}_{a,a',b}} \in \mathsf{RM}[\mathbb{F}, d, 2]$. (The last equivalence follows easily from the first but we need both later on.)

4.3 The "Plane-vs.-Plane" tester

The converse to Problem 4.2 is known as a "low-degree test" (LDT) and by now a number of LDTs are known (cf. Arora et al. [1998]; Polishchuk and Spielman [1994]; Rubinfeld and Sudan [1996]; Arora and Sudan [2003]). The following one is due to Raz and Safra [1997] (a more complete analysis appears in Moshkovitz and Raz [2008]). In what follows we define $\mathsf{agree}(f, g) \triangleq 1 - \mathsf{rHamm}(f, g)$ to be the agreement between two functions $f, g : D \to \Sigma$, and for S a set of functions with domain D and range Σ we define $\mathsf{agree}(f, S) = \max_{g \in S} \mathsf{agree}(f, g)$.

Theorem 4.2. *For integers m, d and finite field $\mathbb{F}$ there exists $\varepsilon_0 = \mathrm{poly}(md/|\mathbb{F}|)$ such that for every function $f : \mathbb{F}^m \to \mathbb{F}$ we have*

$$\mathsf{agree}(f, \mathsf{RM}[\mathbb{F}, d, m]) \geq \mathbb{E}_{a,a',b}\left[\mathsf{agree}(f|_{\mathsf{plane}_{a,a',b}}, \mathsf{RM}[\mathbb{F}, d, 2])\right] - \varepsilon_0$$

Problem 4.3. What are the basic LTC parameters — proximity parameter, query complexity, and soundness — of RM-codes that can be deduced from the previous Theorem?

We will not provide the full proof of the Theorem here. Rather, for the remainder of this session we focus on a key lemma in the proof, stated next. Compared to Theorem 4.2, in the lemma we fix m to the simplest nontrivial value 3, and the right hand side is quadratically smaller.

Lemma 4.3 (Base-case)**.** *For integer d and finite field $\mathbb{F}$ there exists $\varepsilon_0 = \mathrm{poly}(d/|\mathbb{F}|)$ such that for $f : \mathbb{F}^3 \to \mathbb{F}$*

$$\mathsf{agree}(f, \mathsf{RM}[\mathbb{F}, d, 3]) \geq \left(\mathbb{E}_{a,a',b}\left[\mathsf{agree}(f|_{\mathsf{plane}_{a,a',b}}, \mathsf{RM}[\mathbb{F}, d, 2])\right]\right)^2 - \varepsilon_0$$

Let Planes be the set of planes in $\mathbb{F}^3$. A *plane oracle* is a function $\mathbb{O} : \mathsf{Planes} \to \mathsf{RM}[\mathbb{F}, d, 2]$ which assigns to each plane in $\mathbb{F}^3$ a degree-d bivariate polynomial. For $\mathsf{plane}, \mathsf{plane}'$ two non-parallel planes (which intersect at a line line) we say plane and plane' are *consistent*, denoted $\mathbb{O}(\mathsf{plane}) \equiv \mathbb{O}(\mathsf{plane}')$, if the restriction of $\mathbb{O}(\mathsf{plane})$ to line agrees with the restriction of $\mathbb{O}(\mathsf{plane}')$ to line.

Problem 4.4. Prove: If

$$\mathbb{E}_{a,a',b}\left[\mathsf{agree}(f|_{\mathsf{plane}_{a,a',b}}, \mathsf{RM}[\mathbb{F}, d, 2])\right] \geq \gamma$$

Then there exists a plane oracle $\mathbb{O}$ such that

$$\Pr_{\mathsf{plane},\mathsf{plane}'}\left[\mathbb{O}(\mathsf{plane}) \equiv \mathbb{O}(\mathsf{plane}')\right] \geq \gamma^2 - \frac{d+1}{|\mathbb{F}|}.$$

Hints: Define $\mathbb{O} = \mathbb{O}_f$ to be the oracle which assigns to plane the polynomial that has maximal agreement with $f|_{\mathsf{plane}}$, breaking ties arbitrarily. Compare the probabilities of the following three experiments:

1. Pick a random point $x \in \mathbb{F}^3$ and then a random plane plane passing through it; define an indicator random variable Z_x for the event "$\mathbb{O}_f(\mathsf{plane})$ agrees with f on x".
2. Pick a random point $x \in \mathbb{F}^3$ and then two planes $\mathsf{plane}, \mathsf{plane}'$ passing through it; define an indicator random variable Z'_x for the event "$\mathbb{O}_f(\mathsf{plane})$ agrees with f on x and $\mathbb{O}_f(\mathsf{plane}')$ agrees with f on x".
3. Pick two random planes $\mathsf{plane}, \mathsf{plane}'$; define the indicator random variable for the event $\mathbb{O}_f(\mathsf{plane}) \equiv \mathbb{O}_f(\mathsf{plane}')$.

You may use the inequality $\mathbb{E}\left[X^2\right] \geq (\mathbb{E}\left[X\right])^2$ which holds for any real-valued random variable X.

From here on we analyze $\mathbb{O}$ using its *consistency graph*

$$G_{\mathbb{O}} \triangleq \left(V_{\mathbb{O}} = \mathsf{Planes}, E_{\mathbb{O}} = \{(\mathsf{plane}, \mathsf{plane}') | \mathbb{O}(\mathsf{plane}) \equiv \mathbb{O}(\mathsf{plane}') \text{ or } \mathsf{plane} \cap \mathsf{plane}' = \emptyset\}\right)$$

Problem 4.5. Prove: For every non-edge $(\mathsf{plane}, \mathsf{plane}') \notin E_{\mathbb{O}}$,

$$\Pr_{\mathsf{plane}''}\left[(\mathsf{plane}, \mathsf{plane}'') \in E_{\mathbb{O}} \text{ and } (\mathsf{plane}', \mathsf{plane}'') \in E_{\mathbb{O}}\right] \leq \frac{d+1}{|\mathbb{F}|}$$

Hints: By symmetry, assume wlog that $\mathsf{line} = \mathsf{plane} \cap \mathsf{plane}' = \{(x, 0, 0) | x \in \mathbb{F}\}$. Bound the probability of each of these events: *(i)* $\mathsf{plane}'' \cap \mathsf{line} = \emptyset$, and *(ii)* $\mathsf{plane}'' \cap \mathsf{line} \neq \emptyset$ *and* plane'' agrees with both plane and plane' on their common intersection.

Problem 4.6. Let G be a graph with the following property: For any non-edge (u, v), the number of common neighbors of u and v is at most $\varepsilon|V|$. (Our consistency graph satisfies this with $\varepsilon = \frac{d+1}{|\mathbb{F}|}$.) Prove that one can remove $O(\sqrt{\varepsilon}|V|^2)$ edges from V and partition the residual graph into singletons and cliques of size $\geq 2\sqrt{\varepsilon}|V|$. Hints: analyze the following process:

1. While there exists a vertex v with less than $2\sqrt{\varepsilon}|V|$ neighbors, remove all its incident edgs; If no such v exists then
2. A vertex u is *active* if *(i)* u is not isolated (i.e., has at least 1 edge) and *(ii)* u's connected component is not a clique. If no u is active then terminate; Else,

3. Pick active u arbitrarily; remove all edges between the neighbors of u and the vertices at distance 2 from u; go back to step 1.

Bound the number of edges removed in steps 1, 3. Argue that upon termination all vertices are either singletons, or members of large cliques.

Problem 4.7. Complete the proof of Lemma 4.3 using the Schwartz-Zippel Lemma (Problem 4.1).

Lecture 5

Arithmetization using low degree polynomials

November 26 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

Recall that the Hadamard-based, exponential length PCP construction leading to Theorem 3.1 had two essential parts, an LTC tester for the Hadamard code, and a reduction from satisfiability to LTC testing. In Theorem 4.2 we proved the analog of Theorem 2.4, and we now focus on reducing satisfiability to low-degree testing. In what follows an *m-to-m' polynomial map* $\vec{P} : \mathbb{F}^m \to \mathbb{F}^{m'}$ is a sequence of m' polynomials $P_1, \ldots, P_{m'} \in \mathbb{F}[X_1, \ldots, X_m]$ and the degree of $\vec{P}$ is defined to be the maximal degree of $P_1, \ldots, P_{m'}$.

Definition 5.1 (Algebraic Constraint Satisfaction Problem (ACSP))**.** An instance of ACSP is a tuple $\psi = (m, \mathbb{F}, H, \vec{P}_1, \ldots, \vec{P}_k, Q_1, \ldots, Q_a)$ satisfying

- m is an integer
- $\mathbb{F}$ is a finite field and $H \subset \mathbb{F}$
- $\vec{P}_1, \ldots, \vec{P}_k$ is a sequence of m-to-m polynomial maps
- $Q_1, \ldots, Q_a \in \mathbb{F}[X_1, \ldots, X_m, Y_1, \ldots, Y_k]$

An *assignment* to ψ is a polynomial $A \in \mathbb{F}[X_1, \ldots, X_m]$. We say it *satisfies* ψ iff

$$\forall \vec{x} = (x_1, \ldots, x_m) \in H^m, \quad Q_i\left(\vec{x}, A\left(\vec{L}_1(\vec{x})\right), \ldots, A\left(\vec{L}_k(\vec{x})\right)\right) = 0 \tag{2}$$

For every $i = 1, \ldots, a$. We say ψ is satisfiable iff there exists A that satisfies it. The language ACSP-SAT contains all satisfiable ACSP instances.

We will now show two reductions: *(i)* From circuit-SAT (Definition 3.3) to ACSP-SAT, and *(ii)* from ACSP-SAT to low-degree testing. Combining the two (and picking parameters in a good way) will lead to a PCP system with polynomial-length proofs, poly-logarithmic query complexity, perfect completeness and constant positive soundness (cf. Theorem 5.4).

5.1 Arithmetization: From circuit-SAT to ACSP-SAT

The first reduction is captured by the following theorem.

Theorem 5.2 (Reduction to ACSP)**.** *There exists a polynomial time reduction from circuit-SAT to ACSP where a circuit ϕ with n gates is reduced to $\psi = (m, \mathbb{F}, H, \vec{P}_1, \ldots, \vec{P}_3, Q_1, Q_2)$ satisfying:*

- $|H|^m = \Theta(n)$

- *The individual degree of each* $\vec{P}_i$ *is at most* $|H|$
- *The individual degree of* Q_i *in* x_j *is* $|H|$ *and the degree in* Y_1, Y_2, Y_3 *is* $O(1)$.
- *If* ψ *is satisfiable then it is satisfied by* $A \in \mathbb{F}[X_1, \ldots, X_m]$ *of individual degree at most* $|H|$.

Problem 5.1. Prove the following Theorem 5.2. Hints:

- Pick arbitrary $H \subset \mathbb{F}$ and associate the n gates of ϕ with H^m arbitrarily.
- Let $p_2 : H^m \to H^m$ define the map that sends a gate-index i to its first input j. Similarly, define p_3 to be the map that sends i to its second input k (if it exists). Let $\vec{P}_2, \vec{P}_3$ be the low-degree extension (or interpolation) of p_2, p_3.
- Let $\hat{Q}_1(Y_1)$ be the polynomial that vanishes iff its input is $\{0,1\}$-valued.
- Let $\hat{Q}_{\text{AND}}(Y_1, Y_2, Y_3)$ be the polynomial that vanishes iff Y_1 is the AND of Y_2, Y_3, assuming all three variables are $\{0,1\}$-valued. Similarly define $\hat{Q}_{\text{NOT}}$ to be the "constraint" that checks a NOT gate.
- Use multivariate Lagrange interpolation to check $\hat{Q}_1$ for every $\vec{x} \in H^m$. This will be Q_1. Similarly use Lagrange interpolation to check the right constraint for each gate. (How do you check that the last gate evaluates to 1?)

5.2 Zero-testing: From ACSP-SAT to low-degree testing

The second reduction is given by the next theorem, due to Alon and Tarsi [1999].

Theorem 5.3 (Combinatorial Nullstellensatz)**.** *Let* $Q \in \mathbb{F}[X_1, \ldots, X_m]$ *have individual degree* d_i *in variable* X_i*. Let* $H \subset \mathbb{F}$ *be of size* $h = |H|$*. Let* $\mathsf{Split}_H(Z) = \prod_{\alpha \in H}(Z - \alpha)$*. Then*

$$\forall \vec{x} = (x_1, \ldots, x_m) \in H^m \quad Q(\vec{x}) = 0$$

if and only if there exist $Q_1, \ldots, Q_m \in \mathbb{F}[X_1, \ldots, X_m]$ *such that*

$$Q(X_1, \ldots, X_m) = \sum_{i=1}^{m} Q_i(X_1, \ldots, X_m) \cdot \mathsf{Split}_H(X_i) \tag{3}$$

$$\deg_j(Q_i) \leq \begin{cases} \max\{0, d_j - h\} & j \leq i \\ d_i & \text{otherwise} \end{cases}$$

Problem 5.2. Prove Theorem 5.3. One direction is easy. For the harder direction

- Start with $m = 1$. Consider $Q(X_1) \mod \mathsf{Split}_H(X)$ and write

$$Q(X_1) = \mathsf{Split}_H(X_1) \cdot Q_1(X_1) + R(X_1)$$

- What are the degrees of Q_1 and R?
- Prove that $R \equiv 0$.
- Use induction to prove the Theorem for general m.

5.3 A polynomial length, polylogarithmic query, PCP Theorem

Combining Theorem 4.2, Theorem 5.2 and Theorem 5.3 gives

Theorem 5.4 (PCP with polynomial length proofs and polylogarithmic query complexity)**.** *There exists a constant $\rho > 0$ such that for every proper complexity function $T : \mathbb{N}^+ \to \mathbb{N}^+$,*

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} \ell(n) & \leq & \mathrm{poly}(T(n)) & & & \\ t(n) & \leq & \mathrm{poly}(T(n)) & & & \\ r(n) & = & O(\log T(n)) & c & = & 1 \\ q & = & \mathrm{poly}(\log T(n)) & s & \geq & \rho \\ \Sigma & = & \{0,1\} & & & \\ \mathsf{nonadaptive} & & & & & \end{array}\right).$$

The notation for the list of restrictions is the same as in Theorem 1.3.

Problem 5.3. Prove Theorem 5.4. You may assume $\vec{P}_i$ is a 1-to-1 map on $\mathbb{F}^m$. Hints:

- The general strategy is similar to that employed in proof of Theorem 3.1. Soundness is the hardest part, and to argue it we will examine two "bad cases": *(i)* that a proof is not close to being low-degree, and *(ii)* that a proof is low-degree but the relevant polynomials do not satisfy the ACSP.
- Pick $|H| \approx \log n$ and $|F| = \mathrm{poly}(|H|)$ and $m = \log n / \log\log n$.
- Define a proof to be the evaluation of polynomials $A, A_1^{(1)}, \ldots, A_m^{(1)}, A_1^{(2)}, \ldots, A_m^{(2)} \in \mathbb{F}[X_1, \ldots, X_m]$ where A supposedly satisfies Q_1, Q_2 from Theorem 5.2 and the $A^{(1)}$ and $A^{(2)}$ polynomials show that Equation 2 holds according to Theorem 5.3.
- Define a verifier that checks Theorem 4.2, Equation 2 and Equation 3.
- Using Theorem 4.2 argue that if any of the A polynomials is not close to low-degree, the proof will be rejected with probability 99% (What are the query complexity and proximity parameter?)
- If all polynomials are close to low-degree, then use Problem 4.1 to argue a constant rejection probability for the tests corresponding to Equation 2 and Equation 3.

Lecture 6

PCPs of Proximity and Proof Composition

December 17 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

The Hadammard based PCP (Theorem 3.1) has constant query complexity, alphabet size, and soundness but super-polynomial length proofs. The RM-based PCP has polynomial length proofs and constant soundness but polylogarithmic query complexity and alphabet size. We now study two methods for combining the best of both worlds to get polynomial length proofs with constant soundness, alphabet size and query complexity. The methods are *code concatenation* and *proof composition*. We start with an illustration of concatenation, the simpler one.

6.1 Concatenation

Problem 6.1. Suppose $C = \{f : D \to \mathbb{F}_{2^k}\}, D \subseteq \mathbb{F}_{2^k}$ is a linear code of rate ρ over $\mathbb{F}_{2^k}$ — the finite field of size 2^k. Recall that each entry $y = f(x), x \in D$ of a codeword $f \in C$ is an element y of $\mathbb{F}_{2^k}$. Assume $\mathbb{F}_{2^k}$-elements are represented using a basis $b_1, \ldots, b_k \in \mathbb{F}_{2^k}$ that is linearly independent over $\mathbb{F}_2$. We thus get a transformation from $\mathbb{F}_{2^k}$ to $\mathbb{F}_2^k$. *Concatenate* C with the k-dimensional Hadamard code by replacing y by the Had_k encoding of y (cf. Definition 3.2) for each $x \in \mathbb{F}_{2^k}$. Denote the concatenated code by $C \circ \mathsf{Had}_k$.

1. What are the alphabet, blocklength, rate, and relative distance of the concatenated code?

2. Prove: If C is a (q, ε, δ)-LTC over alphabet $\mathbb{F}_{2^k}$, then $C \circ \mathsf{Had}_k$ is a $(q', \varepsilon', \delta')$-LTC over alphabet $\mathbb{F}_2$. What are $q', \varepsilon', \delta'$?

6.2 PCP of Proximity (PCPP)

The idea of proof composition was introduced by Arora and Safra [1998]. The basic idea is as follows. Assume we have a nonadaptive verifier V which on CIRCUIT-SAT instance ϕ of size n makes $q(n)$ queries. Nonadaptivity means we can view V as taking the randomness R and computing from it two things: *(i)* a set of query indices I_R, and *(ii)* a decision predicate Dec_R that decides whether to accept or reject the answers given by the proof oracle. The crucial point is that Dec_R is an instance of CIRCUIT-SAT and typically its size is much smaller than n. So we can request the prover to provide an auxiliary proof for each Dec_R that V wishes to check. And then we *compose* V with an *inner verifier* V_R that checks Dec_R using this auxiliary proof.

The main problem with this approach is that each individual Dec_R is satisfiable, so a malicious prover can cheat by giving answers that are inconsistent with a proof that V would accept for ϕ. One way to solve this problem is to use *PCPs of Proximity*, introduced by Ben-Sasson et al. [2004]; Dinur and Reingold [2004].

Definition 6.1 (PCP of Proximity (PCPP))**.** Let ϕ be an instance of CIRCUIT-SAT with n wires. A $(\ell, t, r, d, q, \Sigma)$-*restricted PCPP verifier* for ϕ is a randomized algorithm V that receives a randomness input $R \in \{0,1\}^r$ and outputs after t steps a sequence of q indices $I_R = (i_1, \ldots, i_q), i_j \in [n + \ell]$ along with a *decision predicate* $\mathsf{Dec}_R : \Sigma^I \to \{\mathsf{accept}, \mathsf{reject}\}$ which is a circuit of size d with inputs in the finite alphabet Σ. The parameter ℓ is called the PCPP length. We say V has *completeness* $c \in [0,1]$ and *soundness function* $s : [0,1] \to [0,1]$ for C if:

- **Completeness:** If $\alpha \in \{0,1\}^n$ satisfies ϕ then there exists $\pi \in \Sigma^\ell$ such that

$$\Pr_{R \in \{0,1\}^r} [\mathsf{Dec}_R ((\alpha \circ \pi)|_{I_R}) = \mathsf{accept}] \geq c$$

  where $(\alpha \circ \pi)|_{I_R} \in \Sigma^{I_R}$ is the restriction of α and π to the query set I_R. (I.e., $(\alpha \circ \pi)$ is a string of length $n + \ell$ over Σ and we assume $\Sigma \supseteq \{0,1\}$.)

- **Soundness:** If $\alpha \in \{0,1\}^n$ does not satisfy ϕ, then for all $\pi \in \Sigma^\ell$

$$\mathbb{E}_{R \in \{0,1\}^r} [\mathsf{rHamm} ((\alpha \circ \pi)|_{I_R}, \mathrm{SAT}(\mathsf{Dec}_R))] \geq s(\mathsf{rHamm}(\alpha, \mathrm{SAT}(\phi))).$$

  where $\mathrm{SAT}(\phi)$ is the set of assignments satisfying ϕ (and $\mathrm{SAT}(\mathsf{Dec}_R)$ is similarly defined). If ϕ is not satisfiable (i.e., $\mathrm{SAT}(\phi) = \emptyset$) we define $\mathsf{rHamm}(\alpha, \mathrm{SAT}(\phi))$ to be 1.

Finally, we say that ϕ has a $(\ell, t, r, d, q, \Sigma, \epsilon)$-PCPP if it has a $(\ell, t, r, d, q, \Sigma)$-restricted PCPP verifier with completeness $c = 1$ and soundness function satisfying $s(\delta) \geq \epsilon \cdot \delta$.

Problem 6.2. Show that PCPPs imply PCPs: Suppose there exists a polynomial time algorithm that for every CIRCUIT-SAT instance ϕ of size n produces a verifier V_ϕ that gives a $(\ell, t, r, d, q, \Sigma, \epsilon)$-PCPP. Then CIRCUIT-SAT $\in \mathbf{PCP}\left(\text{list of restrictions} \;\middle|\; \begin{matrix} c \\ s \end{matrix} \right)$. Fill in the restrictions and the value of c, s as a function of the parameters of the PCPP.

Problem 6.3. Let $C = \{w : [n] \to \mathbb{F}_2\}$ be an $[n, k, d]_{\mathbb{F}_2}$-linear code and ϕ a circuit deciding membership in C, i.e., $w \in C$ iff $\phi(w) = 1$. (Such ϕ can be constructed given the parity check matrix of C.) Suppose ϕ has a $(\ell, t, r, d, q, \Sigma, \epsilon)$-PCPP. Then there exists a (q', ϵ', δ')-LTC that can be constructed from C and the PCPP system. What are the basic coding parameters of the LTC (i.e., alphabet, dimension, blocklength and distance)? What are q', ϵ', δ')?

Problem 6.4. Convert our two PCP theorems — Theorem 3.1, Theorem 5.4 — into PCPP theorems. What are the relevant parameters $\ell, t, r, d, q, \epsilon$?

Problem 6.5. Prove the PCPP composition Theorem: If ϕ of size n has a $(\ell, t, r, d, q, \epsilon)$-PCPP and every ϕ' of size d has a $(\ell', t', r', d', q', \epsilon')$-PCPP, then ϕ has a $(\ell'', t'', r'', d'', q'', \epsilon'')$-PCPP. What are $(\ell'', t'', r'', d'', q'', \epsilon'')$ as a function of $(\ell, t, r, d, q, \epsilon)$ and $(\ell', t', r', d', q', \epsilon')$?

Lecture 7

Soundness boosting via gap amplification

December 31 2013

Lecturer: Eli Ben-Sasson Scribe: Eli Ben-Sasson

Both PCP Theorems we saw so far — Theorem 3.1 and Theorem 5.4 — used algebraic methods to obtain a *gap* between completeness, which thus far has been *perfect* (i.e., $c = 1$), and soundness. The method of PCPP proof composition (Problem 6.5) reduces soundness while reducing query complexity and increasing proof length. We now study *gap amplification* due to Dinur [2007], a method that boosts soundness efficiently. Later on we will see a different method that achieves a similar effect, namely, *parallel repetition* Raz [1998].
Roughly speaking, gap amplification is a method that doubles the size of a CNF while doubling the soundness parameter. In more words, it is reduction that maps a constraint satisfaction problem (CSP) instance ϕ of arity 2 to a different CSP instance ϕ' of arity 2, where *(i)* ϕ' is only c times larger than ϕ where c is an absolute positive constant and size is measured by number of constraints, *(ii)* if ϕ is satisfiable then so is ϕ', and *(iii)* If every assignment falsifies at least an ϵ-fraction of the constraints of ϕ then every assignment falsifies at least a 2ϵ-fraction of ϕ. Formally,

Theorem 7.1 (Gap Amplification Theorem)**.** *There exists a constant size alphabet Σ and constant $s_{\max} > 0$ such that*

$$\mathbf{PCP}\left(\begin{array}{l|l} \ell = \ell_0(n) & \\ q = 2 & \\ \Sigma & c = 1 \\ r = r_0(n) & s = s_0(n) \\ t \le T_0(n) & \\ \text{nonadaptive} & \end{array}\right) \subseteq \mathbf{PCP}\left(\begin{array}{l|l} \ell_1(n) = O(\ell_0(n)) & \\ q = 2 & \\ \Sigma & c = 1 \\ r_1(n) = r_0(n) + O(1) & s(n) \ge \min(2s_0(n), s_{\max}) \\ t \le T_0 + \text{poly}(\ell_0(n))) & \\ \text{nonadaptive} & \end{array}\right).$$

Let us explore a few corollaries of this theorem.

Problem 7.1. Prove the following variant of Theorem 1.4 which obtains polynomial-length PCPs with constant soundness, alphabet size and query complexity: There exists $\epsilon > 0$ such that

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} q & = & 2 & & & \\ \Sigma & = & O(1) & & & \\ t(n) & \le & \text{poly}(T(n)) & c & = & 1 \\ \ell(n) & \le & \text{poly}(T(n)) & s & \ge & \epsilon \\ \text{nonadaptive} & & & & & \end{array}\right)$$

Problem 7.2. Prove the following variant of Theorem 1.3 with quasilinear length proofs and constant soundness and alphabet size (but with non-succinct, polynomial, running time):

There exists $\epsilon > 0$ such that

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} \ell(n) & \leq & T(n)\cdot \mathrm{polylog}(T(n)) & & & \\ t(n) & \leq & \mathrm{poly}(T(n)) & & & \\ r(n) & = & \log(\ell(n)) + O(1) & c & = & 1 \\ q & \leq & 2 & s & \geq & \epsilon \\ \Sigma & = & O(1) & & & \\ \mathsf{nonadaptive} & & & & & \end{array}\right).$$

Hints: Use Theorem 7.1 and the following quasilinear PCP result from Ben-Sasson and Sudan [2005]:

$$\mathbf{NTIME}\left(T(n)\right) \subseteq \mathbf{PCP}\left(\begin{array}{lcl|lcl} \ell(n) & = & T(n)\cdot \mathrm{polylog}(T(n)) & & & \\ q & = & O(1) & c & = & 1 \\ \Sigma & = & \mathbb{F}_2 & s & \geq & \frac{1}{\mathrm{polylog}(T(n))} \\ t(n) & = & T(n)^{O(1)} & & & \end{array}\right).$$

7.1 Gap Amplification Theorem — Proof overview

It will be easier to prove Theorem 7.1 using graph theory, so we restate the theorem in this language. To do this we need the following definition.

Definition 7.2 (Constraint Graphs and their soundness)**.** A *Constraint-Graph* (CG) is a triple $\mathcal{G} = (G, \Sigma, C)$ where $G = (V, E)$ is an undirected graph, with self loops and multiple edges. Σ is an alphabet. C is a set of constraints, one for each edge: $C = \{C_e : \Sigma \times \Sigma \to \{\mathsf{accept}, \mathsf{reject}\} | e \in E\}$.
An *assignment* is a function $A : V \to \Sigma$. The *soundness* of A is the fraction of the constraints that are not satisfied by it,

$$S(A, \mathcal{G}) = \Pr_{e \in E, e(u,v)}[C_e(A(u), A(v)) = \mathsf{reject}].$$

The soundness of $\mathcal{G}$ is is the minimal soundness obtained by an assigment,

$$S(\mathcal{G}) = \min_{A:V\to\Sigma} S(A, \mathcal{G}).$$

For a soundness function $\hat{S} : \mathbb{N}^+ \to [0, 1]$, the language Gap-CG$(\Sigma, S)$ is the promise problem defined by

$$\text{Yes} = \{\mathcal{G} | S(\mathcal{G}) = 0\}$$
$$\text{No} = \{\mathcal{G} | S(\mathcal{G}) \geq \hat{S}(n)\}$$

The following is a restatement of Theorem 7.1 using constraint graphs.

Theorem 7.3 (Theorem 7.1 restated with constraint graphs)**.** *For every sufficiently large* Σ *there exists a constant* $s_{\max} > 0$ *(depending on* $|\Sigma|$*) such that* GAP-CG$(\Sigma, s(n))$ *is reducible in polynomial time to* GAP-CG$(\Sigma, \min\{2s(n), s_{\max}\})$*, where* n *denotes the size of the graph, and the reduction increases graph size only by a constant multiplicative factor.*

7.2 Reduction to expander constraint graphs

There are three steps in the reduction of Theorem 7.3. First, the constraint graph G_0 is converted into a special kind of graph G_1. The main attributes of G_1 are that it has *constant degree*, it is *regular* — all vertices have same number of neighbors, each vertex has a *self-loops*, and, most importantly, it is *expanding* under the following definition.

Definition 7.4 (Expander graphs)**.** Let d be an integer and $\lambda < d$. A (d, λ)*-expander graph* $G = (V, E)$ is a d-regular undirected graph with self-loops for each vertex, which satisfies: For any $F \subset E$ and $i \geq 0$, the probability that a random walk stating at a random edge in F makes its $(i+1)$-step in F is at most $\frac{|F|}{|E|} + \left(\frac{\lambda}{d}\right)^i$.

The following lemma is the first step in the proof of Theorem 7.3, its proof is rather standard and we omit it (cf. Dinur [2007]). It says that by incurring a constant reduction in soundness we can assume without loss that our constraint graph is an expander.

Lemma 7.5 (Expanderizing)**.** *For every sufficiently large integer* d *and constant* $\lambda > 2\sqrt{d-1}$ *there exists an absolute constant* $c_1 > 1$ *satisfying the following.* GAP-CG$(\Sigma, s(n))$ *is reducible in polynomial time to* GAP-CG$(\Sigma, s(n)/c_1)$*. Furthermore, a constraint graph* $\mathcal{G}$ *with* n *edges is reduced to a constraint graph* $\mathcal{G}_1$ *over a* (d, λ)*-expander graph with* $O(n)$ *vertices (and edges).*

7.3 Repetition and gap amplification

To discuss the main step in the reduction, the one where soundness in increased, we define the *product* of a constraint graph and its assignment.

Definition 7.6. Let $\mathcal{G} = (G = (V, E), \Sigma, C)$ be a constraint graph and t be an integer. The *product graph* $G^t = (V, E^t)$ has the same vertex set V as that of G, and for every path p of length t in G with endpoints v_0, v_t there is an edge labeled p from v_0 to v_t in E^t. Let $B(v, t)$ denote the ball of radius t around v, i.e., it is the set of vertices of distance at most t from v. An assignment to $\mathcal{G}^t$ is a mapping which assigns to every $v \in V$ a label $\hat{A}(v) \in \Sigma^{B(v,t)}$. The constraint associated with path p (of length t) with endpoints v_0, v_t accepts an assignment $\hat{A}(v_0), \hat{A}(v_1)$ if and only $\hat{A}(v_0)$ and $\hat{A}(v_1)$ agree on the assignment to p and satisfy all constraints of $\mathcal{G}$ that pertain to edges on p.

Lemma 7.7 (Gap amplification)**.** *For every integers* d*, alphabet size* $|\Sigma|$ *and constant* $\lambda < d$ *there exists a constant* β *satisfying the following. If* $\mathcal{G}$ *is a constraint graph over a* (d, λ)*-expander and* t *an integer, then* $S(\mathcal{G}^t) \geq \beta\sqrt{t} \cdot \min\{S(\mathcal{G}), 1/t\}$.

The final part in the proof of Theorem 7.3 reduces the alphabet from $|\Sigma|^{d^t}$ back to a constant, using PCPP composition.

Lemma 7.8 (Alphabet reduction)**.** *For every $\Sigma', |\Sigma'| > |\Sigma|$ there exists a constant $c_3 > 1$ such that* Gap-CG$(\Sigma', s'(n))$ *is reducible to* Gap-CG$(\Sigma, s(n)/c_3)$ *via a polynomial time reduction, which increases graph-size by at most a constant factor.*

Problem 7.3. Prove Theorem 7.3 using the previous three lemmata.

References

1. Noga Alon and M Tarsi. Combinatorial nullstellensatz. *Combinatorics Probability and Computing*, 8(1):7–30, 1999.

2. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. ISSN 0004-5411.

3. Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998.

4. Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

5. László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, New York, NY, USA, 1991. ACM Press. ISBN 0-89791-397-3.

6. László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

7. Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability — towards tight results. *SIAM Journal on Computing*, 27(3):804–915, June 1998.

8. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *STOC*, pages 585–594, 2013a.

9. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from rams to delegatable succinct constraint satisfaction problems: extended abstract. In *ITCS*, pages 401–414, 2013b.

10. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC-04)*, pages 1–10, New York, June 13–15 2004. ACM Press.

11. Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short pcps verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005a.

12. Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3cnf properties are hard to test. *SIAM Journal on Computing*, 35(1):1–21, 2005b.

13. Eli Ben-Sasson and Madhu Sudan. Short PCPs with poly-log rate and query complexity. In *STOC*, pages 266–275, 2005.

14. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 73–83, New York, NY, USA, 1990. ACM Press. ISBN 0-89791-361-2.

15. Dinur. The PCP theorem by gap amplification. *JACM: Journal of the ACM*, 54, 2007.

16. Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP-theorem. In *FOCS*, pages 155–164, 2004. URL http://csdl.computer.org/comp/proceedings/focs/2004/2228/00/22280155abs.htm.

17. Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996. ISSN 0004-5411.

18. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. ISSN 0097-5397.

19. Johan Håstad. Some optimal inapproximability results. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-888-6.

20. F Florence Jessie MacWilliams and NJ Neil James Alexander Sloane. *The Theory of Error-correcting Codes: Part 2*, volume 16. Elsevier, 1977.

21. Thilo Mie. Short pcpps verifiable in polylogarithmic time with $o(1)$ queries. *Ann. Math. Artif. Intell.*, 56(3-4):313–338, 2009.

22. Dana Moshkovitz and Ran Raz. Two-query pcp with subconstant error. *J. ACM*, 57(5), 2010.

23. Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. *SIAM Journal on Computing*, 38(1):140–180, 2008.

24. Ryan O'Donnell. A history of the PCP theorem. Course notes on the PCP Theorem and Hardness of Approximation, Autumn 2005. URL http://www.cs.washington.edu/education/courses/533/05au/pcp-history.pdf.

25. Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 194–203, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-663-8.

26. Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.

27. Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

28. Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.