1 A constant number of query bits

Lecture 7: May 28 and June 4, 2003

Our goal is to prove:

Theorem 1 *For every integer $k \geq 1$ and (error) $0 < e < 1$, there is a positive integer p such that there is a*

$$(k, r = O(n^2), p, c = 2^p, a = 1, e)$$

inner verifier.

For this we shall introduce an encoding scheme that has exponential redundancy in its codewords. This makes the codewords very long compared to their true information content, which we can afford because this encoding is only applied to words of length smaller than $\log n$. The advantage of using this highly redundent code is that in it a lot of information about the encoded word becomes explicit, allowing the inner verifier to test the inner PCP witness by querying only a constant number of bits.

1.1 The Hadamard Code

The encoding function $E : \{0,1\}^n \longrightarrow \{0,1\}^{2^n}$ encodes a string $x \in \{0,1\}$ by a sequence of 2^n bits. Each bit in this sequence $E(X)$ can be indexed by a string $y \in \{0,1\}^n$. The value of the bit indexed by string y is the inner product $\langle x, y \rangle$. Namely, if $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_n$, then $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$, where summation is taken modulo 2.

Proposition 2 *For $x \neq z$, $\Delta(E(x), E(z)) = 1/2$. Hence the Hadamard code has Hamming distance $2^n/2$.*

Proof: Without loss of generality, assume that x and z differ in their first bit. That is, $x_1 = 0$ whereas $z_1 = 1$. Consider all vectors $y \in \{0,1\}^n$, and arrange them in pair, where y and y' are paired together iff they agree on all bits except for the first. Then for every such pair, $\langle x, y \rangle \neq \langle x, y' \rangle$, whereas $\langle z, y \rangle = \langle x, y \rangle$. Hence $E(x)$ and $E(z)$ differ in exactly one of the two locations y and y'. $\square$

It follows that for $z \in \{0,1\}^{2^n}$, if for some x it holds that $\Delta(E(x), z) < 1/4$, then $E(x)$ is the unique closest codeword to z, and then we may think of x as the decoding of z.

There are two dual views of the Hadamard code, based on two different interpretations of $\sum_{i=1}^{n} x_i y_i$.

1. View the x_i as coefficients and the y_i as variables. Then the codeword $E(x)$ can be viewed as the linear function $f_x = \sum_{i=1}^{n} x_i y_i$ evaluated on all possible inputs.

2. View the y_i as coefficients and the x_i as variables. Then the codeword $E(x)$ can be viewed as evaluating all possible linear functions (over $GF(2)^n$) at the point $x \in \{0,1\}^n$.

1.2 The linearity test

Given a string $z \in \{0,1\}^{2^n}$, we would like to test whether it is (close to) a codeword of the Hadamard code. As noted in Section 1.1, valid codewords can be viewed as linear functions f_x. Likewise, we view z as a Boolean function f, and accessing the bit at location $y \in \{0,1\}^n$ can be viewed as getting the value of $f(y)$. For two strings $x, y \in \{0,1\}^n$, let $x \oplus y \in \{0,1\}^n$ denote their bitwise exclusive or.

The linearity test: Choose $r, s \in \{0,1\}^n$ independently at random. Reject if $f(r) \oplus f(s) \neq f(r \oplus s)$.

It can easily be verified that if f is indeed a linear function (or equivalently, a codeword), the test never rejects.

Let $\Delta_L(f)$ denote the distance between f and the linear function (codeword) closest to it. The following theorem is proved in [3, 2].

Theorem 3 *If $\Delta_L \geq \delta$, then the linearity test rejects with probability at least δ.*

We shall prove this theorem in Section 1.8.

1.3 Sampling at a nonrandom location

Assume that we have access to some function f. Let f_x be the linear function closest to f, and assume that $\Delta(f, f_x) = \delta$, where δ is sufficiently small. Then x is the decoding of f. To sample f_x at a random location y, we may simply sample f at the same location, and with probability $1 - \delta$ the answer we get is in fact $f_x(y)$. Here we develop a procedure for sampling f_x at a nonrandom location.

Self correcting probe.

1. Choose $r \in \{0,1\}^n$ at random.
2. Read $f(r)$ and $f(r \oplus y)$.
3. Output $f(r) \oplus f(r \oplus y)$.

Lemma 4 *If for some linear function $f_x : \{0,1\}^n \longrightarrow \{0,1\}$ we have that $\Delta(f, f_x) = \epsilon$, then for every $y \in \{0,1\}^n$, the output of the self correcting probe is equal to $f_x(y)$ with probability at least $1 - 2\epsilon$, where the probability is taken over the choice of r.*

Proof: As r is random, $Pr[f(r) \neq f_x(r)] = \epsilon$. Note also that regardless of the value of y, $r \oplus y$ is also random. Hence $Pr[f(r \oplus y) \neq f_x(r \oplus y)] = \epsilon$. Taking the union bound, there is probability at least $1 - 2\epsilon$ that neither of the two events above happen. But in this case the output is equal to $f_x(y)$. $\square$

1.4 Comparing the inner and outer verifiers

Our inner verifier expects to see k tables, that we denote by $f_1, \ldots, f_k$, where each table f_i is supposedly the encoding $E(X_i)$ for some character X_i that the respective outer verifier would have read. In addition, there is a PCP witness π. The inner verifier needs to check that π contains an encoding of the same $X_1, \ldots, X_k$.

The encoding E that we use for $f_1, \ldots, f_k$ is the Hadamard code. In π we include the Hadamard code of the string X that is the concatenation of $X_1 \ldots X_k$.

Let us assume here that for every $1 \leq i \leq k$, $X_i \in \{0,1\}^{n/k}$, and hence $X \in \{0,1\}^n$. Note that the Hadamard code of X (which is of length 2^n) includes also the Hadamard codes of every X_i (which is of length $2^{n/k}$). To explain this point, let us introduce the following notation. For a string $y \in \{0,1\}^{n/k}$, let us denote by $l_{y,i}$ (the "lift" of y with respect to block i) the n bit string that starts with $(i-1)n/k$ bits that are 0, continues with y, and ends with $(k-i)n/k$ bits that are 0. Then every bit of the Hadamard code of X_i is also a bit of the Hadamard code of X by the following identity $\langle X_i, y \rangle = \langle X, l_{y,i} \rangle$.

Assume that the prover gives us tables $f_1, \ldots, f_k$ and f. How do we check that they are consistent? First, we use the linearity test to test that f is close to being a linear function. Denote this unknown closest linear function by f_x. Now we need to check that f_i and f_x agree. For this we pick a random point $y \in \{0,1\}^{n/k}$. We query $f_i(y)$ and compare it to $f(l_{y,i})$. However, as $l_{y,i}$ is not a random point in $\{0,1\}^n$, we do not know that with high probability $f(l_{y,i}) = f_x(l_{y,i})$. To remedy this problem, we probe $f(l_{y,i})$ using the self correcting probe of Section 1.3.

Repeating the test for every i we have that with high probability, f is indeed (close to) the encoding of the concatenation of the X_i.

Summarizing so far, we have seen that using only a constant number of queries the inner verifier can verify that a table f given as part of an inner PCP witness indeed corresponds to the encoding of the concatination of the answers that the outer prover would give. Hence all that remains to be done at this point is to show that f is the encoding of a satisfying assignment of some 3CNF formula (or Boolean circuit) ϕ_r (with $O((\log\log n)^{O(1)})$ variables) that the outer verifier decided upon based on his random string r.

1.5 Satisfiability and quadratic functions

The outer verifier, based on its random string r, decides on k locations to probe in the outer PCP, and on a 3CNF formula ϕ_r that needs to be satisfied by the n variables whose sequence of values is the concatenation of the k sequences of answer bits from the outer PCP. As a consequence of the use of Hadamard codes and the consistency check of Section 1.4, we are now in the situation that we may assume that the inner prover wrote down the truth table of a function f, and moreover, that this function f is the linear function (Hadamard codeword) defined by the concatination of the answer bits of the outer verifier.

Recall our dual interpretation of the Hadamard codeword f_x, as that of giving the value of every possible linear function on x. We shall now augment the linear code to a quadratic code that gives the value of every quadratic function on input x. But before doing so, let us explain why we are interested in quadratic functions.

The string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ is supposedly a satisfying assignment to some 3CNF formula ϕ_r. We want to check (with high probability) that this is indeed the case. That is, we wish to verify that every clause is satisfied by x. Consider for concreteness a clause $C = (y_1 \vee \bar{y}_2 \vee y_3)$ (and note that what we say is general enough to be applicable to arbitrary clauses). Let us add an auxilairy variable z specifically for this clause. Consider the the following two quadratic functions over GF(2):

$g_1 = y_1 + (1 - y_2) - y_1(1 - y_2) + z.$

$g_2 = 1 - (y_3 + z - y_3 z).$

All additions are done modulo 2, so subtraction is the same as addition and is used only so as to make it clearer how the expressions were derived. Note that $g_2 = 0$ iff at least one of y_3 and z equals 1. Note that $g_1 = 0$ iff $z = y_1 + (1 - y_2) - y_1(1 - y_2)$. Hence $z = 1$ and $g_1 = 0$ iff at least one of y_1 and $\bar{y}_2$ equals 1. It follows that in order for both g_1 and g_2 to be 0, the assignment to the variables y_1, y_2, y_3 must satisfy the clause C.

Let m be the number of clauses in the formula ϕ_r. We add m auxiliary variables $z_1, \ldots, z_m$ and construct $2m$ quadratic functions $g_1, \ldots, g_{2m}$ as above in the varaibles $y_1, \ldots, y_n$ and $z_1, \ldots, z_m$. The formula ϕ_r is satisfied by the assignment x to the y variables, if and only if this asignment can be augmented by an assignment to the z variables such that all $2m$ quadratic functions are 0.

Fix an assignment to all variables (for simplicity, call it x). Assume further that x is encoded using a quadratic code that gives the values of all possible quadratic functions on x. How does one check using a small number of queries that $2m$ specific quadratic functions simultaneously evaluate to 0? We use the same approach that was used in Lecture 3. Pick $2m$ random bits $r_1, \ldots, r_{2m}$ and consider the quadratic function $g_r = \sum_{i=1}^{2m} r_i g_i$. If all $g_i = 0$, then regardless of r, $g_r = 0$. If at least one $g_i = 1$, then with probability $1/2$ over the choice of r, $g_r = 1$. Hence by choosing r at random and reading only one bit from the quadratic code of x, we can determine (with one-sided error probability of $1/2$) whether ϕ is satisfiable.

The number of random bits used by the above procedure is $2m$. In Lecture 3 we considered approaches for reducing the number of random bits used. Here this is unnecessary, because m is very small compared to the number of random bits used by the outer verifier.

1.6 The quadratic code

Given a string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, its quadratic code is simply the list of values of all possible quadratic functions on x. A quadratic function can be speficied using n^2 $0/1$ coefficients, one for each product $x_i x_j$. Note that over GF(2), there is no need to specify coefficients for the linear terms x_i, because $(x_i)^2 = x_i$. We consider only homogeneous quadratic functions, not involving an additive constant. As the functions g_1 and g_2 above do involve an additive constant (which must be 1, as this is the only nonzero constant in GF(2)), we remark here that considering homogeneous functions is not a significant restriction, as the inner verifier may always add the additive constant by himself.

Hence $Q(x)$, the quadratic code of x, is a word containing 2^{n^2} bits. As every linear function is also a quadratic function, part of $Q(x)$ is also the Hadamard code $E(x)$. There is an even stronger connection between linear codes and quadratic codes. For vectors $y = (y_1, \ldots, y_n)$ and $z = (z_1, \ldots, z_n)$, let $y \otimes z$ denote the outer product (or tensor product) of y and z, namely the n^2-bit string giving all products $y_i z_j$. It can easily be verified that the quadratic codeword of x is identical to the linear codeword of $x \otimes x$.

Let $f_1 : \{0,1\}^n \longrightarrow \{0,1\}$ and $f_2 : \{0,1\}^{n^2} \longrightarrow \{0,1\}$ be two functions prepared by the prover. We already showed the linearity test that can be used in order to check that f_1 is close to linear, namely, that for some (unknown) $x \in \{0,1\}^n$, f_1 is close to the linear function f_x. We now wish to check that f_2 is close to the linear function $f_{x \otimes x}$, for the same (unknown) x. We may assume that the linearity test was performed also on f_2, so we do know that f_2 is close to some linear function f_z (where $z \in \{0,1\}^{n^2}$), and it remains to check whether the unknown z equals the unknown $x \otimes x$.

The quadratic consistency test. Pick $u, v \in \{0,1\}^n$ at random. Reject if $f_1(u)f_1(v) \neq f_2(u \otimes v)$.

Proposition 5 *If there is some $x \in \{0,1\}^n$ such that $f_1 = f_x$ and $f_2 = f_{x \otimes x}$ then the quadratic consistency test never rejects.*

Proof: We then have that

$$f_1(u)f_1(v) = \langle x, u \rangle \cdot \langle x, v \rangle = \sum_{i,j} x_i u_i x_j v_j = \langle x \otimes x, u \otimes v \rangle = f_2(u \otimes v)$$

□

Proposition 6 *If there is some $x \in \{0,1\}^n$ and $z \in \{0,1\}^{n^2}$ such that $f_1 = f_x$ and $f_2 = f_z$ and $z \neq x \otimes x$ then the quadratic consistency test rejects with probability at least $1/4$.*

Proof: Let X denote the order n matrix one gets be arranging the vector $x \otimes x$ in a natural order in a matrix (namely, entry ij is $x_i x_j$). Then

$$f_1(u)f_1(v) = \langle x, u \rangle \cdot \langle x, v \rangle = \sum_{i,j} u_i x_i x_j v_j = u^t X v$$

Similarly, arrange z in a matrix Z such that

$$f_2(u \oplus v) = \sum_{ij} Z_{ij} u_i v_j = u^t Z v$$

Now if $Z \neq X$, there is probability at least $1/2$ over the choice of random vector v that $Zv \neq Xv$. Conditioned on inequality holding, there is probability $1/2$ over the choice of the random vector u that $u^t(Zv) \neq u^t(Xv)$. □

Observe that the quadratic consistency test probes f_1 at two random locations, but probes f_2 at a nonrandom location. Hence the test needs to be modified so that f_2 is probed using the self correcting probe. We then have that if $\Delta_L(f_1) \leq \epsilon$ and $\Delta_L(f_2) \leq \epsilon$, and f_2 is not close to the quadratic codeword that corresponds to the same x for which $\Delta(f_1, f_x)$ is minimized, then the quadratic consustency test rejects with probability at least $1/4 - 4\epsilon$.

1.7 Summarizing the constant bit verifier

The outer verifier (which is the combination of outer and inner verifier in the proof that $NP \subset PCP(\log n, (\log\log n)^{O(1)})$ works as follows. Using a random string r it decides on k locations to probe in the outer PCP witness and on a 3CNF formula ϕ_r that the answers should satisfy.

The (truthful) inner prover encodes the contents of every location of the outer PCP witness (each such location contains $(\log\log n)^{O(1)}$ bits) using the Hadamard code. In addition, for every random string r of the outer verifier the inner prover prepares an inner PCP witness π_r. This witness has several components:

1. A table f_1 that is the Hadamard code f_x of the string x, where the string x is the concatenation of the following components:
   (a) The k answers of the outer PCP to the k probes of the outer verifier when the outer verifier is using random string r.
   (b) Values to the auxiliary z variables that are introduced by the quadratic functions that are derived from ϕ_r.
2. A table f_2 that is the quadratic code $Q(x)$ of the same x.

The inner verifier performs the following tests:

1. Linearity tests on f_1 and f_2.
2. Consistency tests between f_1 and the k tables (supposedly, codewords) that correspond to the k answers of the outer prover.
3. Quadratic consistentcy test (self correcting version) between f_1 and f_2.
4. Chooses a random $r' \in \{0,1\}^{2m}$ and checks that the value of the quadratic function $g_{r'}(x) = 0$. This can be done by probing the table f_2 at two locations (one location would have sufficed had we not needed the error correcting probe).

The total number of queries made by the verifier is constant ($O(k)$), and each query reads only one bit. The total number of random bits used by the verifier is $O(\log n)$ for the outer verifier, and a negligible amount of random bits for the inner verifier. True proofs are always accepted. The inner PCP rejects with constant probability if f_1 is not an encoding of a satisfying assignment for ϕ_r. Using the composition theorem of Lecture 5 we conclude that $NP \subset PCP(\log n, 1)$.

1.8 Analysis of the linearity test

We analyse here the linearity test that was presented in Section 1.2. The analysis that we perform uses Fourier analysis, and is essentially taken from [2]. It can be improved when $\Delta_L(f)$ is small (which as we have seen, is often the easy case for low degree tests), but is essentially best possible when $\Delta_L(f)$ is large. This can be seen for random f, where $\Delta_L(f) \simeq 1/2$ and the linearity test rejects with probability roughly 1/2.

For function f, let us denote $\Delta_L(f)$ by ϵ. We note that the range of interest for ϵ is $0 \leq \epsilon \leq 1/2$, as it is not hard to show that for every function f, its expected agreement with a random linear function is on $(2^n \pm 1)/2$ values. Let us denote by rej the probability that the linearity test rejects on f. We need to show that rej $\geq \epsilon$.

f is a vector in $\{0,1\}^{2^n}$. For convenience, with every vector $v \in \{0,1\}^{2^n}$, let us associate a vector $v' \in \{1,-1\}^{2^n}$, be replacing every 1 by (-1) and every 0 by 1. (That is, value b is replaced by $b' = (-1)^b$.) Hence we obtain a vector of Euclidean norm $\sqrt{2^n}$ in R^n. For $1 \leq i \leq 2^n$, let $w_i \in \{0,1\}^{2^n}$ be the 2^n codewords of the Hadamard code. Taking inner products over the reals and using Proposition 2, we see that for $i \neq j$, $\langle w'_i, w'_j \rangle = 0$. Hence the w'_i form an orthogonal basis for R^{2^n}, where each basis vector has norm $\sqrt{2^n}$. We call it here the "Fourier basis", as the process that we are following here is known as the "discrete Fourier transform".

Any vector in R^{2^n} can be expressed as a linear combination of its projection on the basis vectors. In particular, let us express f' over the Fourier basis. We get the Fourier coefficients $2^{-n}\alpha_i = \langle f', w'_i \rangle$ (inner product is taken over the reals), which are the cosines of the angles between f' and w'_i. Hence $f' = \sum_{i=1}^{2^n} \alpha_i w'_i$. A large Fourier coefficient in absolute value means that f' is highly correlated with w'_i The following proposition shows that for every f, there is a strong connection between its Fourier coefficients and $\Delta_L(f)$.

Proposition 7 *Using the notation above, $\alpha_i = 1 - 2\Delta(f, w_i)$. In particular, when $\Delta_L(f) = \epsilon$, then $\alpha_i \leq 1 - 2\epsilon$ for every i.*

Proof: Every location where f and w_i agree contributes 2^{-n} to α_i, and every location where they disagree contributes -2^{-n}. $\square$

There are many known properties of Fourier transforms. We list a few of them that we shall find useful. Rather than index the basis vectors by 1 up to 2^n, let us index them by the n-bit strings that define the respective codewords. Then it is not hard to see that α_{0^n} gives the average value of f'. Likewise, we have:

Proposition 8 *The function f evaluated at the the point 0^n satisfies $f'((0^n)') = \sum_{i=1}^{2^n} \alpha_i$.*

Proof: All Hadamard codewords w_i have 0 at location 0^n, and hence the respective vector w_i' has 1 at this location. $\Box$

An important operation in the context of Fourier transforms is that of a *convolution.* The convolution $f * g$ of two functions $f, g : \{0,1\}^n \longrightarrow \{0,1\}$ is the function $h : \{0,1\}^n \longrightarrow \{0,1\}$ that satisfies $h'(x) = \sum_{y \oplus z = x} f'(y)g'(z)/2^n$. (Note that the convolution operation is defined of f', g' and not on f, g.) It is not hard to verify that the convolution operation is commutative, associative, and distributive with respect to addition.

The reader may observe that the convolution operation is strongly connected to our linearity test. For given x, it averages over all ways of breaking x into an exclusive or of a pair of inputs. The linearity test picks one such pair at random. Getting some intuition from the linearity test, we see that if f and g are highly correlated with some basis vector w_i (have large Fourier coefficient α_i), then we expect their convolution to be still correlated with w_i, though somewhat less. (Pairs (y, z) where both $f(y)$ and $g(z)$ are equal to the respective $w_i(y)$ and $w_i(z)$ contribute positively to this correlation.) This leads to the following well known proposition whose proof is left as homework.

Proposition 9 *Let $\alpha_i^f, \alpha_i^g, \alpha_i^h$ be the respective ith Fourier coefficients of f, g, and $h = f * g$. Then $\alpha_i^h = \alpha_i^f \cdot \alpha_i^g$.*

For a vector $f \in \{0,1\}^n$, the Fourier coefficients of f' satisfy Parseval's indentity, namely, $\sum(\alpha_i)^2 = 1$. (Parseval's identity can be easily proved by taking the inner product of z with itself in two different representations, one using the standard coordinatewise representation, the other using its representation over the Fourier basis.)

Now we complete the analysis of the linearity test for f. Note that for two test points r and s, the expression $(1 - f'(r')f'(s')f'((r \oplus s)'))/2$ evaluates to 1 if the test rejects and to 0 otherwise. Denoting $r \oplus s$ by z, we note that for fixed z the probability of rejecting (over a choice of decomposition of z to $r \oplus s$) is $(1 - (f' * f')(z')f'(z'))/2$. Now ranging over all values of z we in fact get that $\text{rej} = (1 - ((f' * f') * f')(0^n))/2$. Let β_i denote the Fourier coefficients of the function $(f' * f' * f')$. Then by Proposition 8 we have that $\text{rej} = (1 - \sum \beta_i)/2$. By Proposition 9, this gives $\text{rej} = (1 - \sum(\alpha_i)^3)/2$.

Now we note that for all i, $\alpha_i \leq 1 - 2\epsilon$, and moreover $\sum(\alpha_i)^2 = 1$ (by Parseval's indentity). Hence we can conclude that for very small ϵ, $\text{rej} \simeq 3\epsilon$ (up to low order terms), and for arbitrary ϵ

$$\text{rej} = \frac{1 - \sum(\alpha_i)^3}{2} \geq \frac{1 - (1 - 2\epsilon)\sum(\alpha_i)^2}{2} \geq \epsilon$$

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. "Proof verification and hardness of approximation problems". *JACM 45(3): 501–555 (1998).*

[2] M. Bellare, D. Coppersmith, J. Hastad, M. Kiwi, M. Sudan. "Linearity testing in characteristic two". *IEEE Transactions on Information Theory* 42(6):1781–1795, 1996.

[3] M. Blum, M. Luby, R. Rubinfeld. "Self-testing/correcting with applications to numerical problems". *Journal of Computer and System Sciences*, 47(3):549-595, 1993.