Chapter 18

The PCP Theorems

Sections 18.4.2 and 18.4.3 are a bit rough and need work.

The PCP Theorems provide interesting new definitions for the complexity class NP. (Clarification: the singular form "PCP Theorem" will refer to a single result NP = PCP($\log n, 1$) proved in [**?**, **?**], and the plural form "PCP Theorems" refers to a large body of results of a similar ilk, some predating the PCP Theorem.)

Suppose somebody wants to convince you that a boolean formula is satisfiable (or that a certain math statement has a proof, or that a graph has a hamilton cycle, or any of the other NP statements that can be reduced to satisfiability). He could present the usual certificate, namely, a satisfying assignment, which you could then check by substituting into the formula. Can this verification process be made more efficient? Yes, according to the PCP Theorems. He can rewrite his certificate so you can verify it by probabilistically selecting (i.e., using a source of random bits) a constant number of bits —as low as 3 bits— to examine in it. Furthermore, this verification has the following properties: (a) A correct certificate will never fail to convince you (that is, no choice of the random bits will make you reject a correct proof) and (b) An incorrect certificate will convince you with only negligible probability (2^{-100} if you examine 300 bits). In fact, a stronger assertion is true: if the formula is unsatisfiable, then you are guaranteed to reject *any string of letters placed before you* with high probability after examining a constant number of bits. (c) This certificate rewriting is completely mechanical —a computer could do it— and does not greatly increase its size.

Such results have an important consequence: computing good-quality approximate solutions to many NP-hard optimization problems is also NP-hard (in other words, no easier than computing optimum solutions). This consequence, though surprising and mysterious-sounding at first glance, is actually quite natural.

18.1 Probabilistic checking of membership proofs

According to our usual definition, language L is in NP if there is a poly-time Turing machine V ("verifier") that, given input x, checks membership proofs (or certificates) π that $x \in L$:

$$x \in L \Rightarrow \exists \pi . V^{\pi}(x) = 1$$
$$x \notin L \Rightarrow \forall \pi . V^{\pi}(x) = 0,$$

where V^π denotes "a verifier with access to π".

The class PCP is a generalization of this notion in which the verifier is randomized. Also, the verifier has *random access* to the proof string Π. This means that the proof string is accessed via a special *address tape*: if the verifier desires say the ith bit $\Pi[i]$ in the proof string, then it writes i on this tape to receive the bit. Thus a polynomial-time verifier can in principle check membership proofs of exponential size.

An $(r(n), q(n))$-*restricted verifier* is one that uses at most $O(r(n))$ random bits on inputs of size n and queries at most $O(q(n))$ bits of the proof. We denote by $\Pr[V^\Pi(x) = 1]$ the probability that the verifier outputs 1 on proof string Π.

At this point one can distinguish between adaptive and nonadaptive verifiers: a nonadaptive verifier selects its queries based upon just its random string, whereas an adaptive verifier can use bits it has already examined in Π to select which bit position to query next. We assume that the verifier is nonadaptive, and the PCP Theorems below will imply that nonadaptive verifiers are general enough in most situations.

DEFINITION 18.1 *PCP$(r(n), q(n))$ contains every language L for which there is an $(r(n), q(n))$-restricted verifier V such that:*

$$x \in L \Rightarrow \exists \pi. \Pr[V^\pi(x) = 1] = 1$$
$$x \notin L \Rightarrow \forall \pi. \Pr[V^\pi(x) = 0) \geq \frac{1}{2}$$

REMARK 18.1 As usual, the choice of "1" and "1/2" in the definition is somewhat arbitrary. Repeating the verifier k times lowers the error probability from 1/2 to $1/2^k$; however, this also multiplies the number of query bits by k.

18.2 Relation between PCP and standard complexity classes

We make a few simple observations.

1. PCP$(r(n), q(n)) \subseteq$ NTIME$(2^{O(r(n))}q(n))$. PROOF: If V is a verifier for $L \in$ PCP$(r(n), q(n))$, then without loss of generality, we can assume that the proof string π has at most $2^{O(r(n))}q(n)$ bits - since only $O(q(n))$ bits can be queried for any choice of the random string. Instead of performing a PCP verification, the verifier can treat π as a usual witness and simply simulate all possible outcomes of coin flips and calculate the probability of accepting. If the probability is 1 then output "accept" else output "reject". □

2. NP $=$ PCP$(0, poly(n)) = \bigcup_{c>0}$ PCP$(0, n^c)$.

3. *co* RP $=$ PCP$(poly(n), 0)$.

4. GRAPH-NONISOMORPHISM $\in$ PCP$(poly(n), 1)$. PROOF: Say the input to a "yes" instance of GRAPH-NONISOMORPHISM is $x = \{G_1, G_2\}$, where G_1, G_2 have both n nodes. π contains, for each labelled graph H with n nodes, a bit 0/1 corresponding to whether H is isomorphic to G_1 or to G_2 (an arbitrary bit if neither case holds). In other words, π is an array of bits indexed by some encoding of the graph H.

The verifier picks G_1 or G_2 at random, together with a random permutation. She applies the permutation to the vertices of the chosen graph to obtain an isomorphic graph, H, and queries the corresponding bit of π. If the query bit corresponds to the choice of G_1/G_2, then she outputs "success", otherwise "failure".

If G_1 and G_2 are indeed non-isomorphic, then for the π constructed above, every query for which $H \approx G_1$ returns a 0 bit, and every query for which $H \approx G_2$ returns a 1 bit. If G_1 and G_2 are isomorphic, then for any "proof" π the query bit will correspond to the index of the randomly chosen graph with probability $\frac{1}{2}$, as required. □

5. $PERM \in$ PCP($poly(n), poly(n)$). PROOF: Exercise (use the interactive proof for permanent). □

18.2.1 The PCP Theorems

The PCP theorem concerns a surprising new definition of NP.

THEOREM 18.1 (AS '92, ALMSS '92)
$NP =$ *PCP*$(\log n, 1)$.

As mentioned in the chapter introduction, many researchers have proved stronger versions of this theorem. We describe one due to Håstad [**?**] in which the verifier queries only 3 bits from the proof.

Håstad's 3-bit PCP Theorem concerns a $(\log n, 1)$-restricted verifier that works as follows: given an input x and a proof Π it uses its random string r of $O(\log n)$ bits to select three distinct locations $i_1(r), i_2(r), i_3(r)$ in Π, and computes a certain bit b_r. Then it reads the bits $\Pi[i_1(r)], \Pi[i_2(r)], \Pi[i_3(r)]$ and accepts iff

$$\Pi[i_1(r)] + \Pi[i_2(r)] + \Pi[i_3(r)] = b_r \pmod 2 \tag{18.1}$$

THEOREM 18.2
For every $\epsilon > 0$, every NP language L has a verifier of the type described in the previous paragraph, such that if the input $x \in L$, there is a proof Π that the verifier accepts with probability at least $1 - \epsilon$. If the input is not in the language, the verifier accepts every proof with probability at most $1/2 + \epsilon$.

Finally, we note (see exercises) that analogous theorems also hold for NTIME($t(n)$) where poly(n) $\leq t(n) \leq$ exp(poly(n)). For instance, NEXPTIME $= \cup_{c>0}$ PCP($n^c, 1$).

18.3 Approximation algorithms and hardness of approximation

The practical importance of NP-completeness lies in the fact that many combinatorial optimization problems are NP-hard and hence do not have polynomial-time algorithms if P $\neq$ NP. Researchers have therefore tried to design *approximation algorithms*: algorithms that compute near-optimal solutions. The PCP Theorems imply that for many problems, computing near-optimal solutions is no easier than computing exact solutions.

We illustrate the notion of approximation algorithms with an example. MAX-3SAT is the problem of finding, given a 3-CNF boolean formula as input, an assignment that maximizes the number of satisfied clauses. This problem is of course NP-hard, because the corresponding decision problem, 3SAT, is NP-complete.

DEFINITION 18.2 *An algorithm computes an α-approximation to MAX-3SAT if it computes (in polynomial time) for every instance ϕ an assignment that satisfies at least MAX-3SAT(ϕ)/α, where MAX-3SAT(ϕ) is the maximum number of clauses that can be satisfied by any assignment. clauses.*

EXAMPLE 18.1 We describe a polynomial-time algorithm that computes a 2-approximation for MAX-3SAT. We assign values to the variable one by one in a greedy fashion, whereby the ith variable is assigned the value that results in satisfying at least 1/2 the clauses in which it appears. Any clause that gets satisfied is removed and not considered in assigning values to the remaining variables. Clearly, the final assignment will satisfy at least 1/2 of all clauses, and that is certainly within a factor 2 of the maximum that the optimum assignment could satisfy.

Using semidefinite programming one can design an 8/7-approximation to the problem. (See references.)

It was an open question in the early 1990s whether or not there is some $\alpha > 1$ such that computing an α-approximation to MAX-3SAT is NP-hard. Note that in order to prove NP-hardness one needs need a polynomial-time reduction f such that

$$\varphi \in \text{SAT} \Rightarrow \text{MAX-3SAT}(f(\varphi)) = m \tag{18.2}$$

$$\varphi \notin \text{SAT} \Rightarrow \text{MAX-3SAT}(f(\varphi)) < \frac{m}{1+\epsilon}. \tag{18.3}$$

If such a reduction exists, then any $(1+\epsilon)$-approximation algorithm for MAX-3SAT can be used to decide SAT in polynomial time.

EXAMPLE 18.2 We point out that the Cook-Levin theorem does not result in such a reduction. Let L be an NP language and M be an NP Turing machine that recognizes it. Given any input string x, Cook-Levin construct a formula $\varphi_{x,L}$ that is satisfiable iff a valid tableau exists for M on input x in which M accepts.

A tableau is valid iff all 2×3 *windows* look allright, i.e. thay satisfy some local consistency conditions. Cook-Levin reduction encodes these consistency checks with $3CNF$ clauses so that a valid tableau exists iff the set of all these $3CNF$ clauses is satisfiable.

We claim that even if $x \notin L$, we can satisfy almost all the clauses in $\varphi_{x,L}$: just take the tableau which in the first computation step switches to an accepting computation. The consistency conditions are violated only along the first row of the tableau. Since there are $n^{O(1)}$ rows, the fraction of unsatisfied clasues in the corresponding 3SAT formula is $1/n^{O(1)}$, which is much less than any fixed constant ϵ. Thus the Cook-Levin reduction cannot produce a "gap".

Clearly, the problem is that the Cook-Levin theorem uses a fairly nonrobust notion of membership proof, whose correctness relies upon almost every bit being correct. On the

other hand, the PCP Theorem concerns a more robust notion of membership proof for NP, and it implies the existence of the reduction described in (18.2), (18.3). The converse is also true: the existence of such a reduction implies the PCP Theorem (see Exercises). Below, for convenience we will prove a similar —and much stronger—implication of Håstad's 3-bit PCP Theorem, which implies that if P $\neq$ NP then $(8/7-\delta)$-approximation is NP-hard for any $\delta > 0$.

To prove this we define another optimization problem.

DEFINITION 18.3 *Let MAX-E3-LIN be the problem in which the input is a system of linear equations mod 2, each equation contaning at most 3 variables, and the problem is to find an assignment to variables that satisfies maximum number of equations.*

Just as for MAX-3SAT, a simple greedy algorithm computes a 2-approximation for MAX-3LIN-EQN. (In fact, a random 0-1 assignment to the variables will satisfy 1/2 of the equations.) Håstad's 3-bit PCP Theorem implies that if P $\neq$ NP then this is essentially the best one can do.

COROLLARY 18.3
For every $\delta > 0$, computing $(2-\delta)$-approximation to MAX-E3-LIN is NP-hard.

PROOF: Let V be the verifier for SAT as given by Håstad's 3-bit PCP Theorem. Let x be any input. We can construct a system of linear equations from verifier. There is a variable for each bit $\Pi[i]$ in the proof. For each choice of random string r, there is a linear constraint given by (18.1). The set of all these constraints is an instance of MAX-E3-LIN such that depending upon whether or not $x \in$ SAT, either at least $(1-\epsilon)$ fraction of the constraints are satisfiable or at most $(1/2+\epsilon)$ fraction of the constraints are satisfiable. The theorem follows. □

COROLLARY 18.4
For every $\epsilon > 0$, computing $(8/7-\epsilon)$-approximation to MAX-3SAT is NP-hard.

PROOF: We show how to reduce MAX-E3-LIN to MAX-3SAT. Take any instance of MAX-E3-LIN, where we are interested in determining whether $(1-\epsilon)$ fraction of constraints can be satisfied or at most $1/2+\epsilon$ are. Represent each linear constraint by four $3CNF$ clauses in the obvious way. For example, the linear constraint $a+b+c=0 \pmod 2$ is equivalent to the clauses $(\overline{a} \vee b \vee c), (a \vee \overline{b} \vee c), (a \vee b \vee \overline{c}), (\overline{a} \vee \overline{b} \vee \overline{c})$. If a, b, c satisfy the linear constraint, they satisfy all 4 clauses and otherwise they satisfy at most 3 clauses. We conclude that in one case at least $(1-\epsilon)$ fraction of clauses are simultaneously satisfiable, and in the other case at most $1-(\frac{1}{2}-\epsilon)\times\frac{1}{4} = \frac{7}{8}-\frac{\epsilon}{4}$ fraction are. The ratio between the two cases is $\frac{1-\epsilon}{7/8-\epsilon}$, which is $8/7-O(\epsilon)$ as ϵ decreases to 0. Since Håstad's Theorem is true for every constant ϵ, the result follows. □

18.4 Proof of PCP Theorem

To prove the PCP Theorem it suffices to describe a verifier for the language 3SAT. The idea underlying the design of this verifier —as in IP = PSPACE earlier—is an algebraic view

of boolean formulae, which turns the usual "certificate" of satisfiability, namely, a satisfying assignment, into a multivariate polynomial. Many properties of multivariate polynomials can be checked probabilistically by knowing their values at a few random points. At the root of such tests is the simple fact that a nonzero degree d univariate polynomial has only d roots (or the analogous fact about multivariate polynomials, the Schwartz-Zippel lemma).

Polynomials. Let F be a finite field and m, d be integers. An *m-variate polynomial over F* is a sum of terms of the form $ax_1^{j_1}x_2^{j_2}\cdots x_m^{j_m}$, where $a \in F$. The set of all such polynomials is denoted $F[x_1, \ldots, x_m]$. We will often view such a polynomial as a function from F^m to F. The *degree* of the polynomial in its total degree, e.g., $j_1 + \cdots + j_m$ is the degree of the above monomial. Sometimes we also talk about the degree with respect to individual variables; thus the degree of the above monomial with respect to x_1 is j_1. We will usually reserve the symbol q for $|F|$, the cardinality of F.

The *distance* between two functions $f, g : F^m \to F$, denoted $\Delta(f, g)$, is the fraction of points in F^m they differ on. The *agreement* between the functions is $1 - \Delta(f, g)$.

Now we define the algebraic satisfiability problem, which is at the heart of the proof.

Definition 18.4 (Algebraic satisfiability or ALG-SAT) *The input to this problem consists of the following:*

- *a prime number q (specifying a field $GF(q)$),*
- *$p(x_1, x_2, \cdots, x_m, y_1, y_2, \cdots, y_k)$, a polynomial over $GF(q)$ with degree at most d in each x_i and total degree at most 3 with respect to the y_i's.*
- *$\sigma_1, \sigma_2, \cdots, \sigma_k \in S^l$, where $S = GF(q) \cup \{x_1, x_2, \cdots, x_m\}$,*
- *A degree bound D, and*
- *1^n where $n = q^m + q^l$*
- *k, the number of y_i's is $\leq$ poly$(\log n)$.*

This instance is satisfiable *iff there exists a degree D polynomial $B(z_1, z_2, \cdots, z_l)$ such that*

$$p(x_1, x_2, \cdots, x_m, B(\sigma_1), B(\sigma_2), \ldots, B(\sigma_k)) = 0. \tag{18.4}$$

(Note: each σ_i is a vector of l variables or "constants" and hence $B(\sigma_i)$ is welldefined.)

Remark 18.2 The input is padded by 1^n to ensure that the problem is in NP even if we use a very inefficient representation of p and the "witness" B, namely, giving their value at all points.

Example 18.3 Let $p(x_1, x_2, y_1) = x_1^3 + 3x_1x_2^2 + 3x_1^2x_2 + x_2^3 - (x_1 + x_2)y_1$, $h = D = 3$, and $s_1 = (x_1, x_2)$. Then, this instance is algebraically satisfiable with $B(x_1, x_2) = (x_1 + x_2)^2$.

Remark 18.3 Note that there could be two notions of a zero polynomial: the polynomial with all coefficients zero, or a polynomial that is zero at all points. The two notions do not coincide in the finite field $GF(q)$: for example $x^q - x$ is 0 for all $x \in GF(q)$ (by Fermat's little theorem). However, in all our discussions below the field size significantly exceeds the degree and the two notions coincide.

18.4.1 ALG-SAT is NP-complete

As noted, ALG-SAT $\in$ NP. This section proves that ALG-SAT is NP-complete.

THEOREM 18.5
3SAT $\leq_p$ ALG-SAT.

The proof of Theorem 18.5 is rendered more readable if we use the following generalization of ALG-SAT in which the "witness" consists of many polynomials instead of a single polynomial.

DEFINITION 18.5 (MULTIPLE-ALGEBRAIC-SATISFIABILITY OR MALG-SAT) *The input to mALG-SAT problem is similar to ALG-SAT, with the following additions: (a) A bound J on the number of witness polynomials (b) a list $i_1, i_2, \ldots, i_k$ that gives, for each y_j, an index $i_j \in [1, J]$.*

The instance is satisfiable *if there exist J degree-D polynomials $B_1, B_2, \ldots, B_J$ that are l-variate such that*

$$p(x_1, x_2, \cdots, x_m, B_{i_1}(\sigma_1), B_{i_2}(\sigma_2), \ldots, B_{i_k}(\sigma_t)) = 0 \tag{18.5}$$

Clearly, ALG-SAT is a special case of mALG-SAT. Now we show that a "promise" version of mALG-SAT is a special case of ALG-SAT.

DEFINITION 18.6 ("PROMISE MALG-SAT") *In* Promise mALG-SAT *the input is an instance of mALG-SAT as described in Definition 18.5. We are promised that*

- *EITHER there exists a sequence of degree D polynomials satisfying the instance (the input is a YES instance of the promise problem)*
- *OR ELSE no sequence of polynomials of degree even $D + k$ satisfies the instance (the input is a NO instance).*

The goal is to determine which case holds.

LEMMA 18.6
Promise mALG-SAT $\leq_p$ ALG-SAT.

PROOF: Given any instance $\mathcal{I}$ of Promise mALG-SAT as described in Definition 18.6 we take any distinct $u_1, u_2, \ldots, u_k \in F$ and replace every occurence of $B_i(z_1, \ldots, z_l)$ by $B(z_1, \ldots, z_l, u_i)$, where B is some unknown polynomial in $l + 1$ variables. We replace the earlier degree bound of D by $D + k - 1$. We change the padding string 1^n to reflect the new parameters. Call the resulting instance $f(\mathcal{I})$; it is an instance of the ALG-SAT problem since it only involves one unknown polynomial B.

Suppose $\mathcal{I}$ was a YES instance of Promise mALG-SAT. We show $f(\mathcal{I}) \in$ ALG-SAT. Suppose $B_1, B_2, \ldots, B_k$ are the degree D polynomials that satisfy $\mathcal{I}$. We can define

$$B(z_1, z_2, \ldots, z_{l+1}) = \sum_{i=1}^{J} L_i(z_{l+1}) B_i(z_1, \ldots, z_l),$$

where $L_i(x)$ is the degree $k-1$ polynomial that is 1 if $x = u_i$ and 0 if $x \in \{u_1, u_2, \ldots, u_k\} \setminus \{u_i\}$. ($L_i(x)$ is just the Lagrange interpolation polynomial.) Clearly, B has degree at most $D + k - 1$ and it satisfies $f(\mathcal{I})$.

Conversely, suppose $f(\mathcal{I}) \in$ ALG-SAT. Let B be any polynomial of degree $D + k - 1$ that satisfies $f(\mathcal{I})$. Then $B_1, B_2, \ldots, B_k$ satisfy $\mathcal{I}$ where $B_i(z_1, z_2, \ldots, z_l) = B(z_1, \ldots, z_l, u_i)$. But B_i's are polynomials of degree at most $D+k-1$, so $\mathcal{I}$ must actually be a YES instance of Promise mALG-SAT. □

The rest of the section will use a method of encoding a string of field elements of length $(h+1)^m$ by an m-variate polynomial whose degree in each variable is h. (Thus, total degree $\leq mh$.) Since $0, 1$ belong to every field, this method in particular allows encoding every string of $(h+1)^m$ bits by such a polynomial.

LEMMA 18.7 (POLYNOMIAL EXTENSION LEMMA)
Let $H = \{u_0, u_1, \cdots, u_h\}$ be a subset of $h+1$ elements from $GF(q)$. For each function $f : H^m \to F$ there exists a polynomial $\hat{f}(x_1, x_2, \cdots, x_m)$ whose degree in each x_i is h and which satisfies $\hat{f}(v) = f(v)$ for all $v \in H^m$. This polynomial can be computed in time poly(h^m).

PROOF: The requirements are satisfied by the Lagrange interpolating polynomial:

$$\hat{f}(x_1, x_2, \cdots, x_m) = \sum_{v \in H^m} f(u) \Pi_{i=1}^{m} \frac{\Pi_{j \neq i} x_i - u_j}{\Pi_{j \neq i} u_i - u_j} \tag{18.6}$$

□

Now we prove Theorem 18.5. Thanks to Lemma 18.6 it suffices to prove the following.

THEOREM 18.8
3SAT $\leq_p$ Promise mALG-SAT.

PROOF: Let the input to the reduction be φ, a 3CNF formula of size n. The instance of mALG-SAT produced will use a field $F = GF(q)$ where $q = \log^{O(1)} n$. The degree d will be such that $q = d^{O(1)}$.

Let $h\lceil \log n \rceil$ and $H = \{u_0, u_2, \ldots, u_h\}$ be any set of $h+1$ distinct field elements. Assume without loss of generality that φ has $(h+1)^m$ clauses and $(h+1)^k$ variables, where m, k are $O(\log_h n) = O(\frac{\log n}{\log \log n})$.

STEP 1:

The main idea is to interpret clauses as polynomials, namely, the boolean constraint $a_1 \vee \bar{a_2} \vee a_3 = 1$ is interpreted as the algebraic constraint $(1 - a_1)(a_2)(1 - a_3) = 0$ where the boolean values 0/1 are also interpreted as field elements.

Think of a boolean assignment to the variables as a function $A : H^k \to F$. We represent the above constraints more succinctly. Interpreting each $c \in H^m$ as a clause and each $v_i \in H^k$ as a variable, define the function $g^A : H^{m+3k} \to F$ as:

$$g^A(c, v_1, v_2, v_3) = \phi(c, v_1, v_2, v_3) \Pi_{i=1}^{3} (\chi_i(c) - A(v_i)) \tag{18.7}$$

where

$$\phi(c, v_1, v_2, v_3) = \begin{cases} 1 & \text{if each } v_i \text{ is the } i\text{th variable in the clause } c \\ 0 & \text{otherwise} \end{cases} \tag{18.8}$$

$$\chi_i(c) = \begin{cases} 1 & \text{if } v_i \text{ is unnegated in clause } c \\ 0 & \text{otherwise} \end{cases} \tag{18.9}$$

Clearly, a boolean assignment $A : H^k \to F$ satisfies φ iff $g^A(c, v_1, v_2, v_3) = 0$ for all $c \in H^m$ and $v_1, v_2, v_3 \in H^k$. (It is technically possible that A takes nonboolean values to some points in H^k and nevertheless $g^A = 0$, but just considering the boolean values suffices to satisfy the formula.)

In an effort to obtain an instance of mALG-SAT the reduction replaces ϕ, χ_1, χ_2, and χ_3 by their polynomial extensions $\hat{\phi}$, $\hat{\chi_1}$, $\hat{\chi_2}$, and $\hat{\chi_3}$ respectively as given by Lemma 18.7. (The reduction computes these polynomials explicitly in $h^{O(m)} = \text{poly}(n)$ time.) Let $M = m+3k$. Use the symbol C to denote an "unknown" k-variate polynomial. Using the shorthands $\vec{x^0} = (x_1, x_2, \ldots, x_m)$ and $\vec{x^i} = (x_{m+(i-1)k+1}, x_{m+(i-1)k+2}, \ldots, x_{m+ik})$ for $i = 1, 2, 3$, we defind an M-variate polynomial f^C as

$$f^C(\vec{x}) = \hat{\phi}(\vec{x}) \prod_{i=1}^{3} \left(\hat{\chi}_i(\vec{x^0}) - C(\vec{x^i}) \right) \tag{18.10}$$

We say that C is a *satisfying polynomial for* φ iff the restriction of C to H^k, $C|_{H^k} : H^k \to F$, satisfies the formula φ. Clearly, irrespective of the degree of C,

$$C \text{ is a satisfying polynomial iff } \ f^C(\vec{x}) = 0 \ \ \forall \vec{x} \in H^M. \tag{18.11}$$

It may appear that we are now done because of the superficial similarity of condition (18.11) to mALG-SAT, but the similarity is misleading. Since f^C is a polynomial of degree up to $4mh$ (even if C is the constant function), the fact that it is 0 on H^M does not imply that it is identically 0.

STEP 2:

Now we introduce another $M+1$ variables $f_0, f_1, \ldots, f_{M+1}$ that all represent "unknown" M-variate polynomials. We would like to impose the following constraints among them:

$$f_0 = f^C \tag{18.12}$$

$$f_i(x_1, x_2, \ldots, x_M) = \sum_{j=0}^{h} f_{i-1}(x_1, \ldots, x_{i-1}, u_j, x_{i+1}, \ldots, x_M) x_i^j. \tag{18.13}$$

A simple induction shows that the constraints in (18.13) imply

$$f_M(x_1, \ldots, x_M) = \sum_{i_1, i_2, \ldots, i_M \in [0,h]} f_0(u_{i_1}, u_{i_2}, \ldots, u_{i_M}) x_1^{i_1} x_2^{i_2} \cdots x_M^{i_M}. \tag{18.14}$$

This implies that f_M is the zero polynomial iff $f_0 = 0$ on H^M, which by (18.12) happens iff C is a *satisfying polynomial*. Induction also shows that f_i has degree at most $deg(f_0) + h \cdot i$.

STEP 3:

Now we describe a polynomial p in $M+1$ variables that defines an instance of mALG-SAT. The definition of p involves bundling all defining constraints from (18.12), (18.13), as well as the constraint that $f_M = 0$, into a single constraint. Using $\vec{x}$ as shorthand for $(x_1, x_2, \ldots, x_M)$, we define p as

$$p(x_1, x_2, \ldots, x_M, x_{M+1}) = f_M(\vec{x}) + x_{M+1}(f^C(\vec{x}) - f_0(\vec{x})) + \sum_{i=1}^{M} x_{M+1}^{i+1}(f_i(\vec{x}) - \sum_{j=0}^{h} f_{i-1}(x_1, \ldots, x_{i-1}, u_j, x_{i+1}, \ldots, x_M)x_i^j). \tag{18.15}$$

Note that p can be viewed as a univariate polynomial in x_{M+1} whose coefficients are polynomials in $x_1, x_2, \ldots, x_M$. Thus $p = 0$ iff these coefficients are 0, which happens iff the constraints in (18.12), (18.13) are satisfied and $f_M = 0$. Now in the expression for p replace all appearances of the unknown polynomials $C, f_0, \ldots, f_M$ by new variables y_i's as required by the definition of ALG-SAT. Note that p has degree at most 3 with respect to y_i's (only the definition of f^C in (18.10) uses degree 3; otherwise the degree is 1). We let the degree upperbound $D = 4mh + Mh$.

One final sticking point is that C was assumed to be k-variate and $f_0, f_1, \ldots, f_M$ were M-variate, whereas the definition of mALG-SAT requires all unknown polynomials to have the same number of variables. So we just allow C to be an M-variate polynomial, but fix the last $M - k$ coordinates to 0 when using it in p.

Finally, we need to pad the reduction's output by 1^N where $N = q^{M+1}$, but $M = m + 3k = O(\frac{\log n}{\log\log n})$, and $q = (\log n)^{O(1)}$, so $q^{M+1} = \text{poly}(n)$. Hence the padding takes $\text{poly}(n)$ time.

This completes the description of the reduction. Now we briefly argue its correctness. Suppose φ is satisfiable. Then we let C be the degree mh polynomial extension of a satisfying assignment (which makes f^C a degree $4mh$ polynomial) and use (18.12), (18.13) to define $f_0, f_1, \ldots, f_M$. As noted, the degree of these polynomials is at most $deg(f^C) + Mh \leq D$, which is below the degree upperbound. Thus the instance of mALG-SAT is satisfiable.

Now consider the case that φ is unsatisfiable, and $C, f_0, f_1, \ldots, f_M$ are any polynomials of degree at most $D+M+1$ whose substitution makes $p = 0$. But $p = 0$ means $f_0, f_1, \ldots, f_M$ satisfy (18.12), (18.13), which means C is a satisfying polynomial, an impossibility. □

18.4.2 ALG-SAT $\in$ sPCP$(\log n, 1, \log^{O(1)} n)$

Now we define sPCP, a generalization of PCP. The verifiers for languages in this class expect the proof string Π to be "segmented" into chunks of q bits. The verifier is not allowed to read a segment partially: even if it needs one bit in the segment, it has to read the entire segment. Another way to state this restriction is that the proof string is an array of symbols from some alphabet of size 2^q (whose each symbol is thus represented using q bits). Clearly, the usual PCP definition is obtained as a subcase when $q = 1$.

DEFINITION 18.7 (sPCP) *Language L is in sPCP$(r(n), s(n), q(n))$ if it has a verifier that expects the proof string Π to contain segements of length $q(n)$. Using a string w of $O(r(n))$*

random bits the verifier computes a circuit C_w of size poly$(s(n)q(n))$. Then it computes $s = O(s(n))$ indices $i_1, i_2, \ldots, i_s$, reads the corresponding entries $\Pi[i_1], \Pi[i_2], \ldots, \Pi[i_s]$ from the proof string, and accepts iff the string $\Pi[i_1] \circ \Pi[i_2] \circ \cdots \circ \Pi[i_s]$ satisfies C_w. (Here $\circ$ denotes concatenation of strings.) If $x \in L$ then there is a proof string Π that the verifier accepts with probability 1 (over the choice of w) and if $x \notin L$ then the verifier rejects every Π with probability at least $1/2$.

REMARK 18.4 It follows from the definition that $\text{sPCP}(r(n), s(n), q(n)) \subseteq \text{PCP}(r(n), s(n)q(n))$. The exercises ask you to prove a converse: if $q(n) \leq n$ and $r(n) \geq \log n$ then $\text{PCP}(r(n), q(n)) \subseteq \text{sPCP}(r(n), 1, \text{poly}(q(n)))$.

The main result in this section is the following.

THEOREM 18.9
ALG-SAT $\in$ *sPCP*$(\log n, 1, \log^{O(1)} n)$.

Before giving the proof, we describe two algebraic procedures associated with polynomials: *Low degree test* and *Reconstruction procedure*. Both are very query-efficient and work well in the segmented proof model.

Low degree test

The low-degree test is given a function $f : F^m \to F$. Using randomness, it checks that f looks "locally" like a degree-d polynomial. If f passes this local check with reasonable probability, then, surprisingly, one can infer that f has significant agreement with a degree-d polynomial. To describe the "local checks" we need to define *affine subspaces* of F^m.

DEFINITION 18.8 (AFFINE SUBPACES) *A* k-dimensional affine subspace *is a subset of F^m containing $|F|^k$ points with the parametric description*

$$\{\overline{u_0} + t_1\overline{u_1} + t_2\overline{u_2} + \cdots + t_k\overline{u_k} : t_1, t_2, \ldots, t_k \in F\},$$

where $\overline{u_0}, \overline{u_1}, \ldots, \overline{u_k} \in F^m$. Alternatively, we can say that it consists of all affine combinations of a set of $k+1$ vectors, called its affine basis.

REMARK 18.5 Note that the affine subspace is nothing but the usual k-dimensional vector subspace translated by some fixed vector $\overline{u_0}$. Also, its parametric description is not unique —remember that a vector subspace may have multiple bases— but we assume for the rest of the discussion that an affine basis has been fixed for each subspace.

Below, to save words we call a 1-dimensional affine subspace a *line*, a 2-dimensional subspace a *plane* and a 3-dimensional subspace a *cube*.

Notice that if $f : F^m \to F$ is a degree d polynomial, then its restriction on a k-dimensional affine subspace is described by a k-variate degree-d polynomial in the parameters $t_1, t_2, \ldots, t_k$.

EXAMPLE 18.4 The polynomial $x_1^3 x_2 x_3 x_4^2$ is 4-variate and has degree 7. Its restriction on the line $(2, 0, 1, 3) + t(3, 4, 7, 5)$ is the degree 7 univariate polynomial $(2+3t)^3 4t(1+7t)(3+5t)^2$.

The low degree test checks that such a property holds on a randomly chosen 2-dimensional subspace. The test is presented with $f : F^m \to F$, and an integer d. (We allow m, d to be arbitrary, but let us fix them for the rest of the section.) It is also presented a table —called a d-oracle— that is meant to be a "proof" that f is a degree d polynomial. This table contains, for each plane S in F^m, a bivariate degree d polynomial $p_S(t_1, t_2)$ that supposedly describes the restriction of f to that plane. Since we fixed an affine basis for each plane, so for every point $x \in F^m \cap S$, we can associate a pair of values $(t_1, t_2) \in F^2$ and evaluate p_S at that pair. We call the resulting value the *value produced by p_S at x*, denoted $p_S(x)$ —note that this is a slight abuse of notation since x is a point in F^m and p_S is a bivariate polynomial. (We will later use the same notation with lines and cubes as well.)

The Low Degree Test:

Pick a random plane S in F^m and read the associated bivariate polynomial p_S given in the d-oracle.
Randomly pick a point x on plane C and check whether $p_S(x) = f(x)$. If so, ACCEPT, else REJECT.

Clearly, if f has agreement $1-\delta$ with some degree d polynomial, then there is a d-oracle which makes the low degree test accept f with probability at least $1 - \delta$. (Just create the d-oracle by using the polynomial's restriction to all the planes. When the low degree test picks a random point and a random plane containing that point, the probability that the corresponding bivariate polynomial in the d-oracle describes f is just $1 - \delta$.)

The next theorem gives a converse to the above observation.

THEOREM 18.10
There is a $\delta_0 > 0$ (independent of m, d) such that if the low degree test accepts $f : F^m \to F$ with probability at least $1 - \delta$ where $\delta < \delta_0$ then f has agreement $1 - 2\delta$ with some degree d polynomial.

REMARK 18.6 A similar statement is true when one uses random lines instead of random planes. The proof is somewhat simpler for planes, and a lot simpler if one is trying to prove a stronger version of the Theorem, which we will do in the Exercises.

Reconstruction procedure

Suppose a given function $B : F^l \to F$ is δ-close to a degree D polynomial $\tilde{B}$. The polynomial reconstruction procedure reconstructs, with high probability, up to k values of $\tilde{B}$, using a constant number of queries to the proof.

Along the lines of the Low Degree Test, the Polynomial Reconstruction Procedure checks the value of B restricted to a degree k *parameterized curve* $C = \{(g_1(t), g_2(t), \ldots, g_l(t)) \mid t \in F\}$ where $\deg(g_i) \leq k$ for all i. We note that if B is a degree D (multivariate) polynomial, then $B|_C$ is a degree Dk (univariate) polynomial.

The input to the polynomial reconstruction procedure a sequence of k points $\bar{a_1}, \bar{a_2}, \ldots, \bar{a_k}$. The proof string provides tables which evaluate B at every point in F^l. The procedure also

requires the proof string to contain tables for the value of B along each curve whose value at the field elements $u_1, u_2, \ldots, u_k$ is $\bar{a_1}, \bar{a_2}, \ldots, \bar{a_k}$ respectively. Let this function (the table giving the restriction of B to C) be denoted by $g_C(t)$.

The reconstruction procedure works as follows: pick a random curve C, and a random element $t_0 \in F$. Let $\bar{x} = C(t_0)$. Then, if $g_C(t) \neq B(\bar{x})$ output **reject**. Else, output $g_C(u_1), g_C(u_2), \ldots, g_C(u_k)$ as the guesses for $\tilde{B}(\bar{a_1}), \tilde{B}(\bar{a_2}), \ldots, \tilde{B}(\bar{a_k})$.

The procedure works by the following theorem:

THEOREM 18.11
If B is a degree D polynomial, then there exists a proof such that

$$\mathbf{Pr}[\textit{procedure outputs } B(\bar{a_1}), B(\bar{a_2}), \ldots, B(\bar{a_k})] = 1.$$

If B is δ-close to a degree D polynomial $\tilde{B}$, then

$$\mathbf{Pr}[\textit{procedure outputs } \textbf{reject} \textit{ or } \tilde{B}(\bar{a_1}), \tilde{B}(\bar{a_2}), \ldots, \tilde{B}(\bar{a_k})] \;\geq\; 1 - \sqrt{\delta} - o(1).$$

PROOF:

(We only give a sketch of the proof here.)

Since in the case when B is actually degree D the assertion of the theorem is obvious, let us assume that we are in the case when B is δ-close to a degree D polynomial $\tilde{B}$.

Observation: If you randomly pick a curve C (whose value at $u_1, u_2, \ldots, u_k$ is $\bar{a_1}, \bar{a_2}, \ldots, \bar{a_k}$ respectively) then for any fixed $t \in F \setminus \{u_1, u_2, \ldots, u_k\}$, $C(t)$ is a random element of F^l. Therefore, for any fixed $t \in F \setminus \{u_1, u_2, \ldots, u_k\}$,

$$\mathop{\mathbf{Pr}}_{C}[B(C(t) = \tilde{B}(C(t))] = 1 - \delta.$$

Using linearity of expectation, we get

$$\mathop{\mathbf{E}}_{C}[\text{fraction of points on } C \text{ where } B, \tilde{B} \text{ agree}] \;\geq\; 1 - \delta - (1-\delta)\frac{k}{q} \;\geq\; 1 - \delta - \frac{k}{q}.$$

Let $\delta' \doteq \delta + \frac{k}{q}$. By a simple averaging argument, we get

$$\mathop{\mathbf{Pr}}_{C}[B, \tilde{B} \text{ agree on } \geq 1 - \sqrt{\delta'} \text{ fraction of points on } C] \;\geq\; 1 - \sqrt{\delta'}.$$

Thus, with high probability, B and $\tilde{B}$ agree on $1 - \sqrt{\delta'}$ fraction of points of C. In the remaining analysis we will assume that we have picked a curve C in which this actually happens.

Now there are two cases:

- CASE 1: $g_C(t) \neq \tilde{B}|_C$.
  Then,

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{t_0}[g_C(t_0) = \tilde{B}(C(t_0))] \;\leq\; \frac{Dk}{q} \qquad \text{(Schwartz-Zippel)} \\
\implies &\mathop{\mathbf{Pr}}_{t_0}[g_C(t_0) = B(C(t_0))] \;\leq\; \sqrt{\delta} + \frac{Dk}{q} \\
\implies &\mathop{\mathbf{Pr}}_{t_0}[\text{procedure outputs } \textbf{reject}] \;\geq\; 1 - \sqrt{\delta'} - \frac{Dk}{q}.
\end{aligned}$$

- CASE 2: $g_C(t) = \tilde{B}|_C$.
  In this case,

$$\Pr_{t_0}[g_C(t_0) \neq B(C(t_0))] = \Pr_{t_0}[\tilde{B}(C(t_0)) \neq B(C(t_0))] \leq \sqrt{\delta'}.$$

  Therefore,

$$\Pr_{t_0}[\text{procedure outputs } \textbf{reject}] \geq 1 - \sqrt{\delta'}.$$

  If $g_C(t_0) = B(C(t_0))$, the procedure correctly outputs $\tilde{B}(\bar{a_1}), \tilde{B}(\bar{a_2}), \ldots, \tilde{B}(\bar{a_k})$.

□

Proof of Theorem 18.9

PROOF: (currently a sketch) Let p be the polynomial in the given instance of ALG-SAT. We require the proof to contain the table of values of the unknown degree D polynomial B as well as tables for low degree test and polynomial reconstruction. Do a low degree test and verify with high confidence that the provided table for B is δ-close to some degree D polynomial, say $\tilde{B}$.

Pick up random elements $a_1, a_2, \ldots, a_m \in F$, and set $x_i = a_i$ for $1 \leq i \leq m$. Since the total degree of p in $y_1, y_2, \ldots, y_k$ is at most 3, after fixing the values of the x_i what remains is degree 3 polynomial. This polynomial has at most $O(k^3) = \text{polylog}(n)$ terms and hence can be evaluated by a $\text{polylog}(n)$ size circuit for any values of $y_1, y_2, \ldots, y_k$. We try to evaluate p for these values of x_i's, using the provided table for B. This requires reconstructing k values of $\tilde{B}$, which is done using the Polynomial Reconstruction Procedure.

Both the Low Degree Test and the Polynomial Reconstruction Procedure require only a constant number of queries to the segmented proof. Hence, we have ALG-SAT $\in$ sPCP$(\log n, 1, \log^{O(1)} n)$.
□

18.4.3 ALG-SAT $\in$ PCP$(\log^c n, 1)$: Constant-bit verifier

In this section, we will prove the following theorem:

THEOREM 18.12

$$sPCP((\log n)^{O(1)}, O(1), q(n)) \subseteq PCP(\log n + q(n)^{O(1)}, 1).$$

This theorem implies, in conjunction with Theorems 18.5 and 18.9, that NP $\subseteq$ PCP$((\log n)^{O(1)}, 1)$. Before we give the proof of this theorem, we need a few preliminaries.

DEFINITION 18.9 (LINEAR FUNCTION CODE) *The* Linear Function Code *over $GF(2)$ encodes strings of n bits $(a_1, a_2, \ldots, a_n)$ by the linear function*

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} a_i x_i.$$

This function is specified in the proof string by its values at all the points in $GF(2)^n$.

REMARK 18.7 As an encoding method, this is fairly inefficient: n bits are encoded using 2^n bits.

The analog of the Low Degree Test for Linear Function Codes is the following *Linearity Test:*
Given: $f : GF(2)^n \longrightarrow GF(2)$. Pick $\bar{x}, \bar{y}$ randomly from $GF(2)^n$ and accept iff

$$f(\bar{x} + \bar{y}) = f(\bar{x}) + f(\bar{y}).$$

The following theorem, due to Blum, Luby and Rubinfeld (1990) tells us why the test works. It is proved later in Section 18.6.2.

THEOREM 18.13
If $\mathbf{Pr}[\textit{test accepts}] \geq 1 - \delta$ *then* f *agrees with a linear function on* $1 - 2\delta$ *fraction of points.*

The analog of Schwartz-Zippel for linear functions is:

THEOREM 18.14
If $(a_1, a_2, \ldots, a_n) \neq \bar{0}$ *then*

$$\mathop{\mathbf{Pr}}_{\bar{x} \in GF(2)^n}[\sum a_i x_i = 0] = \frac{1}{2}.$$

Linear function reconstruction procedure

Suppose we are given a function $f : GF(2)^n \longrightarrow GF(2)$ which is δ-close to a linear function $\tilde{f}$. Suppose also we are given a point $\bar{z} \in GF(2)^n$ and that we wish to find the value of $\tilde{f}(\bar{z})$.

The procedure to do this is as follows:
Choose $\bar{y} \in GF(2)^n$ randomly and output $f(\bar{y}) + f(\bar{z} + \bar{y})$.

Then, with probability at least $1 - 2\delta$ we get the correct answer: any of $f(\bar{y}), f(\bar{z} + \bar{y})$ can be wrong with probability at most δ, so we can apply the union bound to get the claimed probability of correctness.

Converting a circuit to a system of quadratic equations

THEOREM 18.15
There is a polytime transformation that given any circuit C *with* n *inputs and* m *wires, produces a system of* m *simultaneous bivariate quadratic equations over* $GF(2)$*, in the variables* $x_1, x_2, \ldots, x_n$ *(the input to the circuit) and* $y_1, y_2, \ldots, y_m$ *such that for every* $\bar{a} \in GF(2)$*,*

$$C(\bar{a}) = 1 \Leftrightarrow \exists \bar{b} \in GF(2)^m \textit{ such that } (\bar{a}, \bar{b}) \textit{ satisfies all the equations.}$$

PROOF: The transformation is fairly straightforward: we simply write quadratic equations for each wire. For instance, the equation for $y_i = y_j \vee \bar{y_k}$ is

$$y_i = 1 - (1 - y_j)y_k.$$

It is clear that such equations can be written for all wires. Then for any $\bar{a} \in GF(2)^2$, the corresponding $\bar{b} \in GF(2)^m$ will be just the logical value of the corresponding wires in C when it is fed the input $\bar{a}$. □

Proof of Theorem (18.12)

For convenience, we restate Theorem (18.12):

$$\text{sPCP}((\log n)^{O(1)}, O(1), q(n)) \subseteq \text{PCP}(\log n + q(n)^{O(1)}, 1).$$

PROOF: Let $L \in$ sPCP$((\log n)^{O(1)}, O(1), q(n))$. Then the verifier expects the proof Π to be segmented into chunks of $O(q(n))$ bits each. It builds a circuit C of size $m =$ poly$(q(n))$ using the random string given to it, queries $O(1)$ of the proof chunks, and accepts iff the circuit accepts the concatenation of the proof chunks it queries.

We will now build up a PCP$(\log n + q(n)^{O(1)}, 1)$ verifier for L in several steps. Essentially, we need to check whether the circuit C accepts a $O(1) \times \text{poly}(q(n)) = \text{poly}(q(n))$ size (concatenated) input $\bar{a}$. Let the size of $\bar{a}$ be N.

For an input $\bar{a}$, let $\bar{b} \in GF(2)^m$ be the assignment of wires from theorem (18.15) which proves that $C(\bar{a}) = 1$. Let $(c_1, c_2, \ldots, c_M) = (\bar{a}, \bar{b})$ (thus, $M = N+m$). The verifier requires that the proof should contain linear function encodings of $\bar{c}$ and $\bar{c} \otimes \bar{c}$. $\bar{c} \otimes \bar{c}$ is an M^2 bit string where for $1 \leq i, j \leq M$ the $(i,j)^{\text{th}}$ bit is $c_i c_j$.

Note that though we have assumed here that we have the linear function encodings of $\bar{a}$ (and its derivatives $\bar{b}, \bar{c}, \bar{c} \otimes \bar{c}$), though we do not have direct access to them - we simply have the linear function encodings of $O(1)$ proof chunks of poly$(q(n))$ size. This can be done because we can use the encodings of the proof chunks to *construct* the linear function encodings of the desired bit strings.

The remainder of the proof will be in several steps:

- STEP 1: Using the linearity test, verify that the encodings for $\bar{c}$ and $\bar{c} \otimes \bar{c}$ close to linear functions.

- STEP 2: Let the encoding for $\bar{c}$ be close to the linear function $\tilde{f}_1 = \sum d_i x_i$ and let the encoding for $\bar{c} \otimes \bar{c}$ be close to the linear function $\tilde{f}_2 = \sum d_{ij} x_{ij}$. Then in this step, we check that

  $$d_{ij} = d_i d_j, \quad \forall i, j.$$

  To do this, we choose $\bar{x}, \bar{y} \in GF(2)^M$ at random and verify that

  $$\tilde{f}_1(\bar{x}) \cdot \tilde{f}_1(\bar{y}) = \tilde{f}_2(\bar{x} \otimes \bar{y})$$

  (and reject the proof and halt otherwise). Note that we use the reconstruction procedure for linear functions in this step.

  CLAIM 18.16 *If there exist i, j such that $d_{ij} \neq d_i d_j$ then*

  $$\mathbf{Pr}[\textit{test rejects}] \geq \frac{1}{4}.$$

  PROOF: Consider two matrices $M_1 \doteq ((d_i d_j))$ and $M_2 \doteq ((d_{ij}))$.

  If $M_1 \neq M_2$ then

  $$\mathop{\mathbf{Pr}}_{\bar{x}}[\bar{x}^T M_1 \neq \bar{x}^T M_2] \geq \frac{1}{2}$$

(using Schwartz-Zippel for linear functions for each column).

If $\bar{x}^T M_1 \neq \bar{x}^T M_2$ then

$$\mathop{\mathbf{Pr}}_{\bar{x},\bar{y}}[\bar{x}^T M_1 \bar{y} \neq \bar{x}^T M_2 \bar{y}] \geq \frac{1}{2}$$

(using Schwartz-Zippel again).

Since $\bar{x}, \bar{y}$ are chosen independently at random, we get the desired result. □

- STEP 3: Verify that $(d_1, d_2, \ldots, d_N) = (a_1, a_2, \ldots, a_N)$.

  Pick $\bar{x} \in GF(2)^n$ at random and verify that

  $$\tilde{f}_1(\bar{x}) = \tilde{f}_2(\bar{x}, \underbrace{0, 0, \ldots, 0}_{m})$$

  (and reject the proof and halt otherwise).

- STEP 4: Now the only thing left to do is to check whether the system of m simultaneous bivariate quadratic equations for the circuit C is satisfied by $\bar{c}$.

  Write the equations as polynomials that have to be made zero and let $p_1, p_2, \ldots, p_m$ be these polynomials.

  Pick random $(r_1, r_2, \ldots, r_m) \in GF(2)^m$ and verify that $(d_1, d_2, \ldots, d_M)$ satisfy

  $$\sum_{i=1}^{m} r_i p_i = 0$$

  (and reject the proof and halt otherwise).

  If any p_i is not made 0 after substituting $\bar{d}$ then with probability at least $\frac{1}{2}$ the sum is not 0.

  To perform the test, we notice that the LHS of the above equation is some expression of the form (constant bit + sum of a subset of the d_i + sum of a subset of the $d_i d_j$). This value is essentially (constant + 1 evaluation of $\tilde{f}_2$) and can thus be obtained by querying the proof.

Now we calculate the randomness requirement and number of query bits for the verifier. The randomness requirement is $O(\log n + M) = O(\log n + q(n)^{O(1)})$. The number of query bits is constant in all the steps. Thus, the verifier is a $\text{PCP}(\log n + q(n)^{O(1)}, 1)$ machine, and so we are done. □

18.4.4 NP ⊆ PCP$(\log n, 1)$: sketch

To be written

18.5 Low Degree Test: Correctness

This section proves Theorem 18.10.

Clearly, it suffices to restrict attention to the *optimum* d-oracle, which maximizes the probability of acceptance. This d-oracle contains, for each plane S, the bivariate degree d polynomial that has the highest agreement with f on points of S (if the best polynomial is not unique, ties are broken arbitrarily). Let us call this the *plane polynomial* for S.

Our proof requires that the field size $q = |F|$ is at least $(dm/\delta)^c$ for some constant c, which we will not specify. This affords a chance to simplify notation: we will use $o(\delta)$ to bundle any terms of the type $1/q$, $\sqrt{d/q}$ etc., and the lowerbound on q implies that these are indeed $o(\delta)$. The sum of two $o(\delta)$ terms will be written as $o(\delta)$ again, and we will be careful to do this only $O(1)$ times.

The proof will use symmetry-related arguments, which are all variations on the following fact (which is a simple exercise): if $k < l$, then all k-dimensional affine subspaces are contained in the same number of l-dimensional subspaces. We give a simple example how this is useful.

EXAMPLE 18.5 (*Symmetry-based argument*) Suppose $f : F^m \to F$ passes the low degree test with probability β. For any cube C let y_C denote the accept probability when the test is done with a random plane and point chosen from C. If C is a random cube in F^m then $E_C[y_C] = \beta$. The reason is that this expectation counts each (plane, point) pair in F^m equally often.

REMARK 18.8 The symmetry-based argument is essentially from Arora's Ph.D. thesis, and was sharpened and improved in a paper of Arora and Sudan. The proof of the base case when $m = 3$ is due to Raz and Safra. A more complete history of the low degree test appears in the chapter notes.

18.5.1 The case $m = 3$

In this section we assume $m = 3$, and rely on the simple geometry of planes in F^3: every pair of distinct planes either intersect in a line or are parallel (one is obtained from the other by *shifting* each point by a fixed displacement vector). Only $q - 1$ planes are parallel to a given plane. Furthermore, if l is a line, then $1 - 1/q$ fraction of all planes intersect it. (See exercises.) The rest of the section proves the following theorem.

THEOREM 18.17
If $f : F^3 \to F$ passes the test with probability $1-\delta$ then it has agreement at least $1-3\delta-o(\delta)$ with some degree d polynomial.

In the optimum d-oracle for f, every plane S has an associated *plane polynomial* p_S. We define a graph $G = (V, E)$, whose vertices correspond to planes and two vertices v_1, v_2 are connected by an edge in E iff either (a) the corresponding planes are parallel or (b) the two planes intersect in a line and furthermore, their plane polynomials have the same restriction on this line. In the rest of this section, "vertex" refers to a vertex in G, namely, a plane in F^3 and "point" refers to elements of F^3.

The next Lemma shows that G is a fairly dense graph.

LEMMA 18.18
$|E| \geq (1 - \frac{2\delta}{1-d/q})\binom{|V|}{2}$

PROOF: Let Z be the set of all unordered pairs of vertices $\{v_1, v_2\}$ that correspond to pairs of nonparallel planes. Suppose γ fraction of them do not occur in E. We show that the failure probability of the low degree test is at least $\gamma(1-d/q)/2$, which implies the lemma since the failure probability is known to be δ.

Consider the following test. Randomly pick a pair of nonparallel planes S_1, S_2. Then S_1, S_2 intersect in a line, say l. Pick a random point x on l and randomly pick one of S_1, S_2. Then do the low degree test with that plane and x. (That is, accept iff the plane polynomial produces the value $f(x)$ at x.) A moment's thought shows that this is just the low degree test with some extraneous random bits.

However, by assumption the probability is γ that the above test picks a pair of planes corresponding to a nonedge in E. If this happens, the restrictions of their plane polynomials P_1 and P_2 to l are distinct degree d univariate polynomials, and hence agree on at most d/q fraction of points of l. Let $A \subseteq l$ be this set. Then with further probability $(1-d/q)$ the random point x lies in $l \setminus A$. Then at least one of planes P_1, P_2 must disagree with f on x, and the test picks that plane with probability $1/2$. We conclude that the rejection probability of the test is at least $\gamma(1-d/q)/2$. □

Call a triple of vertices (v_1, v_2, v_3) *nontransitive* iff exactly two of the $\binom{3}{2} = 3$ possible edges between them appear in E. A subgraph of G is *transitive* if it contains no nontransitive triples. Note that a transitive subgraph consists of vertex-disjoint cliques (easy exercise). The next Lemma is the heart of the proof.

LEMMA 18.19
G can be made transitive by removing at most $2\sqrt{\frac{d+1}{q}}\,|V|^2$ edges.

PROOF: First we need a simple observation. For any nonedge (v_1, v_2), let $\beta(v_1, v_2)$ denote the fraction of vertices v_3 such that (v_1, v_2, v_3) is a nontransitive triple. (Notice, this means both (v_1, v_3) and (v_2, v_3) are edges.) Let $\beta_{\max}$ denote the maximum value of $\beta(\cdot)$ for any nonedge.

CLAIM: $\beta_{\max} \leq \frac{d+1}{q}$.
Proof: Suppose (v_1, v_2) is a nonedge for which $\beta_{\max}$ is attained. Then the corresponding planes, say S_1, S_2, intersect in a line l. Since (v_1, v_2) is a nonedge, the restrictions of their plane polynomials P_1 and P_2 to l are distinct degree d univariate polynomials, and hence agree on at most d/q fraction of points of l. Let $A \subseteq l$ be this set.

Consider any vertex v_3 that has edges to both v_1, v_2. Either the corresponding plane S_3 does not intersect l at all—but this happens only for $1/q$ fraction of vertices v_3— or else it intersects l, but $S_3 \cap l$ is a point in A. By symmetry, every point on l has the same number of planes passing through it, so the fraction of such v_3 is at most d/q. We have thus shown $\beta(v_1, v_2) \leq d/q + 1/q$ and the claim is proved.

Now we prove the Lemma. We describe an iterative process that removes edges at each step, and stops only when remaining edges form a transitive subgraph. Let $G_i = (V, E_i)$ be the graph remaining at the start of the ith step, with $G_1 = G$. Let $N_i(v)$ denote the neighboring vertices of v in E_i. The following two rules are used to obtain G_{i+1} from G_i.

- If there is a vertex v_i that has at most $\sqrt{\beta_{\max}}|V|$ neighbors in E_i, then remove all edges incident to it and thus obtain E_{i+1}.
- Otherwise, every vertex has degree at least $\sqrt{\beta_{\max}}|V|$. If G_i is not transitive, there exists vertex v_i whose connected component C_i is larger than $N_i(v_i)$. In this case remove all edges in E_i lying between $C_i \setminus N_i(v_i)$ and $N_i(v_i)$; this gives E_{i+1}.

To prove the Lemma, we show that at most $2\sqrt{\beta_{\max}}|V|^2$ edges get removed.

Clearly, the total number of edges removed due to the first rule is at most $\sqrt{\beta_{\max}}|V|^2$, since the rule can be applied at most $|V|$ times, each time removing at most $\sqrt{\beta_{\max}}|V|$ edges.

Estimate the effect of the second rule as follows. Suppose it was applied in the ith step and let $T_i = C_i \setminus N_i(v_i) \times N_i(v_i)$; thus E_{i+1} contains no edge from T_i. Potentially, all of T_i could be edges in E_i and hence eliminated at step i. However, we now show $|E_i \setminus E_{i+1}| \leq \sqrt{\beta_{\max}}|T_i|$, which implies (since T_i's for different steps are distinct) that at most $\sqrt{\beta_{\max}}$ fraction of all $\binom{|V|}{2}$ potential edges get eliminated by the second rule.

Suppose $(v, v') \in E_i$ where $v \in C_i \setminus N_i(v_i)$ and $v' \in N_i(v_i)$. Then the triple (v_i, v', v) is nontransitive due to the nonedge (v_i, v). For every v the number of $v' \in V$ such that (v_i, v', v) is nontransitive is at most $\beta_{\max}|V|$, so the number of edges removed at the step $|E_i \setminus E_{i+1}| \leq \beta_{\max}|V| \cdot |C_i \setminus N_i(v_i)|$. However, i had degree at least $\sqrt{\beta_{\max}}|V|$ so $|T_i| = |N_i(v_i)| \cdot |C_i \setminus N_i(v_i)| \geq \sqrt{\beta_{\max}}|V| \cdot |C_i \setminus N_i(v_i)|$. Thus we have shown $|E_i \setminus E_{i+1}| \leq \sqrt{\beta_{\max}} \cdot |T_i|$, and the proof is completed. □

Theorem 18.17 can now be proved. Lemma 18.19, together with Lemma 18.18, implies that that G has a subgraph with at least $(1 - \frac{2\delta}{1-d/q} - 4\sqrt{\frac{d+1}{q}})\binom{|V|}{2} \geq (1 - 2\delta - o(\delta))\binom{|V|}{2}$ edges that is a union of vertex-disjoint cliques. Hence G must have a clique W of size at least $(1 - 2\delta - o(\delta))|V|$ (easy exercise). The next exercise shows that every clique corresponds to a degree d polynomial.

Exercise 1 (Interpolation) If $W \subseteq V$ is a clique, then there is a degree d polynomial $\phi_W : F^3 \to F$ whose restriction on each plane in W is exactly the corresponding plane polynomial.

Let $\phi_W : F^3 \to F$ be the polynomial corresponding to clique W. Imagine doing a modified low degree test using, instead of the plane polynomials as before, the restrictions of ϕ_W to the planes. Clearly, its accept probability is just the amount of agreement between f and ϕ_W. However, the accept probability of the true low degree test could be higher by at most $2\delta + o(\delta)$ (namely, when the test picks a plane that is not in W). We conclude that f must have agreement at least $(1 - \delta - 2\delta - o(\delta)) = (1 - 3\delta - o(\delta))$ with ϕ_W, and Theorem 18.17 has been proved.

18.5.2 The Bootstrapping

Now we prove the correctness of the low degree test for general m by "bootstrapping" from the subcase $m = 3$. Let us fix $f : F^m \to F$. The general idea is that if f passes the test with high probability, then symmetry considerations imply that in most cubes, the restriction of f to the cube passes the low degree test with high probability. Hence by our correctness

proof for $m = 3$ we know that for most cubes there is a trivariate polynomial that has good agreement with the restriction of f to the cube. This implies some very strong conditions on f (see Lemmas 18.21 and 18.22) which in turn imply that f has good agreement with a degree d polynomial.

DEFINITION 18.10 *Let $\hat{f} : F^m \to F$ be the function*

$$\hat{f}(x) = \textit{most popular value of } p_S(x) \textit{ among all planes } S \ni x \quad \textit{(break ties arbitrarily)} \tag{18.16}$$

LEMMA 18.20
f and $\hat{f}$ agree in $1 - 2\delta$ fraction of points.

PROOF: By assumption about f, $\Pr_{x,S\ni x}[f(x) = p_S(x)] = 1-\delta$, where S is a random plane containing x.

Let $p = \Pr_x[f(x) \neq \hat{f}(x)]$. If x is such that $f(x) \neq \hat{f}(x)$ then a majority of the planes S passing through x are such that $p_S(x) \neq f(x)$. Hence

$$p \cdot \frac{1}{2} + (1 - p) \geq 1 - \delta,$$

and it follows that $p \leq 2\delta$. Thus the lemma is proved. □

By assumption, $\Pr[p_S(x) = f(x)] = 1 - \delta$ for a random x and plane $S \ni x$. Since δ is small, averaging shows that for most x, and most planes $S \ni x$, the following is true (a) $\hat{f}(x) = p_S(x)$ and (b) p_S describes f in most points of S. The next lemma shows that we can replace "most x" in the previous sentence by "every x."

Remark: The following averaging argument is used repeatedly below: if $a_1, a_2, \ldots, \in [0, 1]$ are numbers whose average is $1 - \gamma$ for a small γ, then at least $1 - \sqrt{\gamma}$ of them are at least $1 - \sqrt{\gamma}$.

LEMMA 18.21
For every $x \in F^m$, at least $1 - 10\sqrt[4]{\delta} - o(\delta)$ fraction of planes $S \ni x$ satisfy (a) $p_S(x) = \hat{f}(x)$ and (b) p_S describes f on at least $1 - 10\sqrt[4]{\delta} - o(\delta)$ fraction of points in S.

PROOF: As in Example 18.5, for a cube $C \subseteq F^m$ let y_C denote the success probability of the low degree test when restricted to C. We claim that for every $x \in F^m$, $E_{C\ni x}[y_C] \geq 1 - \delta - o(\delta)$. The reason is that x together with any plane S such that $x \notin S$ determines a unique cube. So the expectation counts every plane not containing x the same number of times. Thus the expectation is the same –upto a $o(\delta)$ term—as the overall success probability of the low degree test. The reason for the $o(\delta)$ term is that planes containing x are counted differently, but the fraction of planes containing x is $o(\delta)$ (actually $1/q^{m-2}$) so their contribution to the success probability of the low degree test is $o(\delta)$ anyway.

Now averaging shows that $1-\sqrt{\delta}-o(\delta)$ fraction of cubes $C \ni x$, satisfy $y_C > 1-\sqrt{\delta}-o(\delta)$. For such a C, Theorem 18.17 proves that there is a polynomial $g_C : F^3 \to F$ that has agreement $1 - 3\sqrt{\delta} - o(\delta)$ with f. For a plane $S \subseteq C$ let z_S be the fraction of points in S where f and g_C agree. Then $E_{S\subseteq C;S\ni x}[z_S] \geq 1 - 3\sqrt{\delta} - o(\delta)$, since symmetry implies that the expectation gets equal contribution from each point in $C \setminus \{x\}$. We conclude that for

at least $1-2\sqrt[4]{\delta}-o(\delta)$ fraction of planes $S \subseteq C$ containing x, we have $z_S \geq 1-2\sqrt[4]{\delta}-o(\delta)$. If $z_S \geq 1-2\sqrt[4]{\delta}-o(\delta)$ then the restriction of g_C to S describes f in $1-2\sqrt[4]{\delta}-o(\delta)$ fraction of points and hence must be the plane polynomial p_S. In particular, $p_S(x) = g_C(x)$. Thus all such p_S's agree at x.

Now imagine the following test, which we call TEST 1: pick a random plane $S \ni x$ and another random point y outside S. These determine a unique cube C that is randomly distributed among cubes $C \ni x$. Pick a random plane $S_1 \ni x$ distinct from S in C. The test is a success if S, S_1 are such that $p_S(x) = p_{S_1}(x)$ and p_S, p_{S_1} both agree with f in at least $1-2\sqrt[4]{\delta}-o(\delta)$ fraction of their points. The previous paragraph proves that the success probability of TEST 1 is at least $(1-\sqrt{\delta}-o(\delta))(1-4\sqrt[4]{\delta}-o(\delta)) \geq (1-5\sqrt[4]{\delta}-o(\delta))$. (The reason is that the test is a success if C is a type of cube mentioned in the previous paragraph, which happens with probability $(1-\sqrt{\delta}-o(\delta))$ and then both S, S_1 are planes of the type mentioned there.)

Now imagine the following test, which we call TEST 2: pick two random planes $S, S_1 \ni x$ in F^m. Let an *inbetween plane* be some plane that intersects both S, S_1 in a line. (Here is a proof that such a plane exists: Suppose S, S_1 are spanned by (x, u_1, u_2) and (x, v_1, v_2) respectively. Then the plane spanned by (x, u_1, v_1) intersects S in the line (x, u_1) and S_1 in the line (x, v_1).) Pick a random inbetween plane S_2. We say TEST 2 is a success iff all of S, S_1, S_2 are such that $p_S(x) = p_{S_1}(x) = p_{S_2}(x)$, and each of p_S, p_{S_1}, p_{S_2} agree with f in at least $(1-2\sqrt[4]{\delta}-o(\delta))$ fraction of their points.

Note that S, S_2 is a random pair of intersecting planes containing x so they together fit in a random cube containing x. The same is true for S_1, S_2. Thus TEST 2 just consists of two random trials of TEST 1. The trials are not independent, but nevertheless the union bound shows that the probability both succeed is at least $1-10\sqrt[4]{\delta}-o(\delta)$.

Now the Lemma has basically been proved, since it concerns a weaker statement about a single random plane S passing through x. To finish we only have to show that the value $p_S(x)$ produced by most planes at x is $\hat{f}(x)$. It is a simple exercise (see the Problems) to check that if a random pair of planes S, S_1 passing through x satisfy $p_S(x) = p_{S_1}(x)$ with probability at least $1-10\sqrt[4]{\delta}-o(\delta)$, then actually there is a common value that must be produced by at least $1-10\sqrt[4]{\delta}-o(\delta)$ fraction of them. Clearly, such a value can only be $\hat{f}(x)$ (by definition of $\hat{f}$). □

We say that a plane passing through a point $x \in F^m$ is *good for* x if it has the two properties mentioned in the statement of Lemma 18.21. We say that a cube $C \ni x$ is *favorable for* x if among all planes of C that pass through x, at least $1-4\sqrt[8]{\delta}-o(\delta)$ fraction are good for x.

LEMMA 18.22
For every line l and point $b \in l$, there is a cube C containing l that is favorable for b as well as for $1-4\sqrt[8]{\delta}-o(\delta)$ fraction of points in l

PROOF: Consider a random cube C containing l, and of all planes of C that pass through b let z_C denote the fraction that are good for b. We show that

$$E_{C:l\subseteq C}[z_C] = 1-10\sqrt[4]{\delta}-o(\delta). \tag{18.17}$$

Picking C randomly involves picking a random plane in F^m that passes through b but does not contain l. However, $1-o(\delta)$ fraction of planes through b do not contain l, so we can

essentially pick a random plane passing through b. Hence our usual symmetry argument —together with Lemma 18.21— implies (18.17).

Now averaging shows that for every $x \in l$, at least $1 - 4\sqrt[8]{\delta} - o(\delta)$ fraction of cubes C containing l are favorable for x. (Aside: In case the reader is wondering, "$4\sqrt[8]{\delta}$" is the square root of $10\sqrt[4]{\delta}$!) Thus averaging implies that there *exists* a cube that is favorable for $1 - 4\sqrt[8]{\delta} - o(\delta)$ fraction of $x \in l$ (including b), and the Lemma is proved. □

Now we prove that $\hat{f}$ is a degree d polynomial. To do this it suffices to prove the following lemma (see exercises).

LEMMA 18.23
On every line, $\hat{f}$ is described by a degree d univariate polynomial.

PROOF: Let l be a line and let h_l be the best univariate polynomial describing the values of $\hat{f}$ on it. Let b be any point on l. We show that $h_l(b) = \hat{f}(b)$ (here we have our usual abuse of notation, whereby $h_l(b)$ is meant to stand for the value produced by h_l at b). If this is true for every b, the lemma has been proved.

Let C be the cube whose existence is proved in Lemma 18.22 for line l.

Consider the probability with which the low degree test succeeds using planes in C. All but $o(\delta)$ fraction of planes of C intersect l, so it suffices to look at the success probability among those planes. By definition of "good" and "favorable" the success probability of the low degree test among planes intersecting l is at least $(1 - 4\sqrt[8]{\delta} - o(\delta))(1 - 10\sqrt[4]{\delta} - o(\delta))$, which is at least $1 - 5\sqrt[8]{\delta} - o(\delta)$ if δ is a small enough constant.

By the trivariate case of the low degree test, the restriction of f to C has agreement at least $1 - 15\sqrt[8]{\delta} - o(\delta)$ with some trivariate degree d polynomial, say g_C.
CLAIM *If C is favorable for $x \in l$, then $\hat{f}(x) = g_C(x)$.*

The lemma follows from the claim, since it implies that the restriction of g_C to l —a univariate degree d polynomial— describes $\hat{f}$ in at least $1-4\sqrt[8]{\delta}-o(\delta)$ fraction of points of l, and so must actually be h_l! Hence $h_l(b) = g_C(b)$, whereas the claim also shows $g_C(b) = \hat{f}(b)$. Thus we have shown $h_l(b) = \hat{f}(b)$ and the lemma is proved.

Let us prove the claim. Let $W \subseteq C \setminus \{x\}$ be the points of C where f and g_C agree. For any plane $S \subseteq C$ let y_S be the fraction of points of S that lie in W. Symmetry considerations imply that for every $x \in l$, $E_{S \ni x}[y_S] = \frac{|W|}{|C|}$. Hence $E_{S \ni x}[y_S] \geq 1 - 15\sqrt[8]{\delta} - o(\delta)$. Thus in at least $1 - 4\sqrt[16]{\delta} - o(\delta)$ fraction of planes $S \ni x$, we have $y_S \geq 1 - 4\sqrt[16]{\delta} - o(\delta)$. Clearly, if $y_S \geq 1-4\sqrt[16]{\delta}-o(\delta)$, then the restriction of g_C to S is a bivariate polynomial that describes f in $1-4\sqrt[16]{\delta}-o(\delta)$ fraction of points of S, and hence must equal the plane polynomial p_S. Thus the value $p_S(x)$ produced by such a plane at x is the same as $g_C(x)$.

Now suppose cube C is favorable for x. By definition of favorable, for almost every plane S passing through x, the value $p_S(x)$ it produces at x is $\hat{f}(x)$. But we have also shown in the previous paragraph that almost every plane produces the value $g_C(x)$ at x. We conclude that $\hat{f}(x) = g_C(x)$ and the Claim is proved. □

18.6 The Fourier transform technique

The Fourier transform technique for constructing and analyzing PCPs was developed in two significant papers of Håstad. The technique involves calculating the maximum acceptance

probability of the verifier using Fourier analysis of the functions presented in the proof string. The technique is delicate enough to give "tight" inapproximability results now known for MAX-CLIQUE, MAX-3SAT, and many other problems.

To introduce the technique we start with a simple example: the linearity test over $GF(2)$. Then we introduce the *Long Code* and how to test for membership in it. These ideas are then used to prove Håstad's 3-bit PCP Theorem, which implies the NP-hardness of computing a $8/7 - \epsilon$ approximation for $MAX - 3SAT$.

18.6.1 The discrete Fourier transform

The discrete Fourier transform is a tool to study functions on the boolean hypercube. In this section, let the hypercube be written as $\{-1,1\}^n$ instead of the more usual $\{0,1\}^n$.

If $f, g : \{-1,1\}^n \to \Re$ are real-valued functions on the boolean hypercube then their *sum* $f+g$ is defined as

$$(f+g)(x) = f(x) + g(x) \tag{18.18}$$

and their *innerproduct* $< f, g >$ is defined as

$$< f, g >= \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} f(x)g(x). \tag{18.19}$$

Consider the following set of 2^n functions that take values in $\{-1,1\}$: for each subset $\alpha \subseteq [1,n]$, the function χ_α is defined as

$$\chi_\alpha(x) = \prod_{i \in \alpha} x_i \qquad \forall x = (x_1, x_2, \ldots, x_n) \in \{-1,1\}^n \tag{18.20}$$

This is an orthonormal set: $< \chi_\alpha, \chi_{\alpha'} >= 0$ if $\alpha \neq \alpha'$ and 1 if $\alpha = \alpha'$. (Simple exercise.) It is a basis because it has 2^n functions, which is the dimension of the vector space. Thus every function $f : \{-1,1\}^n \to \Re$ can be expressed as a linear combination of basis vectors:

$$f = \sum_{\alpha \subseteq [1,n]} \hat{f}_\alpha \cdot \chi_\alpha \tag{18.21}$$

where the *Fourier coefficient* $\hat{f}_\alpha \in \Re$ is (by simple linear algebra) just $< f, \chi_\alpha >$. We call the expression in (18.21) the *Fourier expansion* of f.

LEMMA 18.24
Every two functions $f, g : \{-1,1\}^n \to \Re$ satisfy

1. $< f, g >= \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha$.
2. *(*Parseval's Identity*)* $\sum_\alpha \hat{f}_\alpha^2 \quad = \quad < f, f >$.

PROOF: The second part is a consequence of the first, and to prove the first we note

$$< f, g > = \quad < \sum_\alpha \hat{f}_\alpha \chi_\alpha, \sum_\beta \hat{g}_\beta \chi_\beta > \quad = \sum_{\alpha,\beta} \hat{f}_\alpha \hat{g}_\beta < \chi_\alpha, \chi_\beta > \tag{18.22}$$

which by orthonormality becomes $\sum_\alpha \hat{f}_\alpha \hat{g}_\alpha$. □

EXAMPLE 18.6 We describe the Fourier transforms of a few functions.

1. If $f(x_1, x_2, \ldots, x_n) = x_i$ (a *coordinate function*, to be encountered again soon) then $f = \chi_{\{i\}}$ and so $\hat{f}_{\{i\}} = 1$ and $\hat{f}_\alpha = 0$ for $\alpha \neq \{i\}$.

2. If f is a random boolean function on n bits, then each $\hat{f}_\alpha$ is a random variable that is a sum of 2^n binomial variables (equally likely to be $1, -1$) and hence looks like a normally distributed variable with standard deviation $2^{n/2}$ and mean 0. Thus with high probability, all 2^n Fourier coefficients have values in $[-\frac{\text{poly}(n)}{2^{n/2}}, \frac{\text{poly}(n)}{2^{n/2}}]$.

The connection to PCPs: High level view

In the PCP context we are interested in boolean-valued functions, i.e., those from $GF(2)^n$ to $GF(2)$. We will think of them as functions from $\{-1,1\}^n$ to $\{-1,1\}$, where we have recast $GF(2)$ as $\{-1,1\}$, which is subgroup of $\Re$ with the usual multiplication. (Thus $0 \in GF(2)$ becomes 1, $1 \in GF(2)$ becomes -1, and $+$ over $GF(2)$ turns into multiplication over $\Re$.)

In this worldview, linear functions over $GF(2)$ turn into the Fourier basis functions. To see this, realize that the linear function $\sum_{i=1}^n a_i x_i$ where each $x_i \in GF(2)$ may be thought of as $\sum_{i \in \alpha} x_i$ where where $\alpha \subseteq [1, m]$ is the subset whose characteristic vector is $(a_1, a_2, \ldots, a_n)$ (i.e., $i \in \alpha$ iff $a_i = 1$). The transformation from $GF(2)$ to $\{-1,1\}$ changes this function into $\chi_\alpha(x) = \prod_{i \in \alpha} x_i$.

The connection between PCPs and Fourier transforms rests on our trivial observation that for any real-valued functions f, g,

$$< f, g >= E_{x \in \{-1,1\}^n}[f(x)g(x)] = \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha.$$

In the PCP setting, the expectation will correspond to the acceptance probability of some test performed by the verifier. The above expression relates this acceptance probability to the Fourier coefficients of f, g. We will usually be able to show that if the verifier's test accepts with nontrivial probability, then only "a few" coefficients are significant (contrast this with a random function, all of whose Fourier coefficients are roughly equal).

A large Fourier coefficient has an intuitive meaning in the PCP setting: it corresponds to Fourier basis functions (namely, linear functions over $GF(2)$) that have significant agreement with f, g. This follows from the following trivial Lemma.

LEMMA 18.25
Function $f : \{-1,1\}^m \to \{-1,1\}$ has agreement $1 - \gamma$ with the basis function χ_α iff $\hat{f}_\alpha = 1 - 2\gamma$.

18.6.2 Analysis of the linearity test over $GF(2)$

Recall that the linearity test is provided a function $g : GF(2)^m \to GF(2)$ and has to determine whether f has significant agreement with a linear function. To do this it picks $x, y \in GF(2)^m$ randomly and accepts iff $f(x + y) = f(x) + f(y)$.

Now we rephrase this test using $\{-1,1\}$ instead of $GF(2)$, so linear functions turn into Fourier basis functions.

DEFINITION 18.11 *If $x, y \in \{-1,1\}^m$ then $x \odot y = (x_1y_1, x_2y_2, \ldots, x_my_m)$.*

We note that for any basis function, $\chi_\alpha(x \odot y) = \chi_\alpha(x)\chi_\alpha(y)$.

THEOREM 18.26
If $f : \{-1,1\}^m \to \{-1,1\}$ satisfies $\Pr_{x,y}[f(x \odot y) = f(x)f(y)] \geq 1 - \delta$ then there is some $\alpha \subseteq [1,m]$ such that f has agreement at least $1 - \delta$ with χ_α.

PROOF: Since the test accepts if $f(x \odot y)f(x)f(y) = 1$ and rejects if $f(x \odot y)f(x)f(y) = -1$, the hypothesis may be rephrased as $E_{x,y}[f(x \odot y)f(x)f(y)] \geq 1 - \delta - \delta = 1 - 2\delta$. We show that this implies $\hat{f}_\alpha \geq 1 - 2\delta$ for some $\alpha \subseteq [1,m]$, whence the Theorem follows by Lemma 18.25. (Remark: from now on the proof will not need f to be boolean-valued, though it does need $< f, f >= 1$.)

Expressing f by its Fourier expansion,

$$E_{x,y}[f(x \odot y)f(x)f(y)] = E_{x,y}[(\sum_\alpha \hat{f}_\alpha \chi_\alpha(x \odot y))(\sum_\beta \hat{f}_\beta \chi_\beta(x))(\sum_\gamma \hat{f}_\gamma \chi_\gamma(y))]. \tag{18.23}$$

Since $\chi_\alpha(x \odot y) = \chi_\alpha(x)\chi_\alpha(y)$ this becomes

$$= E_{x,y}[\sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \chi_\alpha(x)\chi_\alpha(y)\chi_\beta(x)\chi_\gamma(y)]. \tag{18.24}$$

Using linearity of expectation:

$$= \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_{x,y}[\chi_\alpha(x)\chi_\alpha(y)\chi_\beta(x)\chi_\gamma(y)] \tag{18.25}$$

$$= \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_x\left[\chi_\alpha(x)\chi_\beta(x)\right] E_y\left[\chi_\alpha(y)\chi_\gamma(y)\right] \qquad (\because x, y \text{ are independent}). \tag{18.26}$$

By orthonormality $E_x[\chi_\alpha(x)\chi_\beta(x)] = 0$ if $\alpha \neq \beta$ and 1 otherwise, so we simplify to

$$= \sum_\alpha \hat{f}_\alpha^3 \tag{18.27}$$

$$\leq (\max_\alpha f_\alpha) \times (\sum_\alpha \hat{f}_\alpha^2). \tag{18.28}$$

Since $\sum_\alpha \hat{f}_\alpha^2 =< f, f >= 1$, this expression is at most $\max_\alpha \{f_\alpha\}$. Hence $\max_\alpha \hat{f}_\alpha \geq 1 - 2\delta$ and the theorem is proved. □

18.6.3 Coordinate functions, Long code and its testing

A *coordinate function* is a function f from $\{-1,1\}^W$ to $\{-1,1\}$ such that for some $w \in W$, $f(x_1, x_2, \ldots, x_W) = x_w$; in other words, $f = \chi_{\{w\}}$.

DEFINITION 18.12 (LONG CODE) *The* long code *for* $[1..W]$ *involves representing each* $w \in [1..W]$ *by the table of all values of the function* $\chi_{\{w\}} : \{-1,1\}^W \to \{-1,1\}$.

REMARK 18.9 Note that w, normally written using $\log W$ bits, is being represented using a table of 2^W bits, a doubly exponential blowup! This inefficiency is the reason for calling the code "long."

The Long code test, given a function $f : \{-1,1\}^W \to \{-1,1\}$, tries to check if it has good agreement with $\chi_{\{w\}}$ for some w, namely, $\hat{f}_{\{w\}}$ is significant. The test below can only conclude something a bit weaker, namely, that if $\hat{f}_\alpha$ is significant then $|\alpha|$ is small (see Corollary 18.28) but this weaker conclusion suffices in Håstad's PCP Theorem.

Let $\rho > 0$ be some arbitrarily small constant. The test picks random vectors $x, y \in \{-1,1\}^W$ and then a vector $z \in \{-1,1\}^W$ that is -1 in a random subset of coordinates, where each coordinate is picked with probability ρ. Thus whp, about ρ fraction of coordinates in z are -1 and the other $1-\rho$ fraction are 1. We think of z as a "noise" vector. The test accepts iff $f(x)f(y) = f(x \odot y \odot z)$. Note that the test is similar to the linearity test except for the use of the noise vector z.

Suppose $f = \chi_{\{w\}}$. Then

$$f(x)f(y)f(x \odot y \odot z) = x_w y_w (x \odot y \odot z)_w \tag{18.29}$$

$$= x_w y_w (x_w y_w z_w) \qquad = z_w \tag{18.30}$$

Hence the test accepts iff $z_w = 1$ which happens with probability $1-\rho$. Now we prove a certain converse.

LEMMA 18.27
If the test accepts with probability $1/2+\delta$ *then* $\sum_\alpha \hat{f}_\alpha^3 (1-2\rho)^{|\alpha|} \geq 2\delta$.

REMARK 18.10 This is reminiscent of line (18.27) in the proof of Theorem 18.26, except for the extra factor $(1-2\rho)^{|\alpha|}$. This factor depresses the contribution of $\hat{f}_\alpha$ for large α, allowing us to conclude that the small α's must contribute a lot; see Corollary 18.28.

PROOF: If the test accepts with probability $1/2+\delta$ then $E[f(x)f(y)f(x \odot y \odot z)] = 2\delta$. Replacing f by its Fourier expansion,

$$E_{x,y,z}[f(x)f(y)f(x \odot y \odot z)] = E_{x,y,z}[(\sum_\alpha \hat{f}_\alpha \chi_\alpha(x)) \cdot (\sum_\beta \hat{f}_\beta \chi_\beta(y)) \cdot (\sum_\gamma \hat{f}_\gamma \chi_\gamma(x \odot y \odot z))]$$

$$= E_{x,y,z}[\sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma \chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x)\chi_\gamma(y)\chi_\gamma(z)]$$

$$= \sum_{\alpha,\beta,\gamma} \hat{f}_\alpha \hat{f}_\beta \hat{f}_\gamma E_{x,y,z}[\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x)\chi_\gamma(y)\chi_\gamma(z)].$$

Orthonormality implies the expectation is 0 unless $\alpha = \beta = \gamma$,so this becomes

$$= \sum_\alpha \hat{f}_\alpha^3 E_z[\chi_\alpha(z)] \tag{18.31}$$

Since $E_z[\chi_\alpha(z)] = E_z[\prod_{w\in\alpha} z_w] = \prod_{w\in\alpha} E[z_w] = (1-2\rho)^{|\alpha|}$ this is

$$= \sum_\alpha \hat{f}_\alpha^3 (1-2\rho)^{|\alpha|} \tag{18.32}$$

□

Corollary 18.28
If f passes the long code test with probability $1/2+\delta$ then

$$\sum_{\alpha:|\alpha|\le k} \hat{f}_\alpha^3 \ge 2\delta - \epsilon,$$

where $k = \frac{1}{2\rho}\log\frac{1}{\epsilon}$.

18.6.4 Proof of Håstad's 3-bit PCP Theorem

This section proves Theorem 18.2. Let L be any language. The starting point is the PCP Theorem, which implies the existence of a $\gamma > 0$ and a polynomial-time reduction f from the membership problem in L to MAX-3SAT such that if $x \in L$ then the 3CNF formula $f(x)$ is satisfiable and otherwise MAX-3SAT$(f(x)) < 1-\gamma$. Let $f(x)$ have n variables and m clauses. We give a sequence of three verifiers, and describe their actions given input x.

> **Verifier** V_0: *Given input x, expects the proof string to contain tables T_1, T_2. Table T_1 contains m entries, each containing 3 bits, and table T_2 contains n entries, each 1 bit.*
> *Verifier picks a random clause, say the ith, in $f(x)$ and one of the three variables in it, say y_j. It checks that the three bits in T_1 in position i constitute a satisfying assignment to the clause, and that the bit in position j in table T_2 agrees with the bit that was provided in the clause assignment. If so, V_0 accepts.*

Clearly, the PCP Theorem implies that completeness is 1. Furthermore, we claim that if $x \notin L$, then the verifier accepts every proof with probability at most $1-\gamma/3$. The reason is that table T_2 defines an assignment to $f(x)$, and this assignment can satisfy at most $1-\gamma$ fraction of clauses. Hence when the verifier picks a random clause, with probability at least γ it picks a clause that is not satisfied by the assignment. Now either the entry for this clause in T_1 shows that the clause is not satisfied, in which case the verifier immediately rejects. Or else the clause assignment provided for the clause is inconsistent with the values assigned to one of the variables in T_1, and then the verifier rejects when it picks that variablem, which happens with probability 1/3.

Next, we use Raz's *parallel repetition theorem*, which concerns a new verifier V_{raz} that is the k-wise repetition of V_0. (Here k is a large constant specified later.)

Verifier V_{raz}: *Given input x, expects the proof string to contain tables T_1, T_2. Table T_1 contains m^k entries (one for each k-tuple of clauses in $f(x)$) each containing $3k$ bits, and table T_2 contains n^k entries (one for each k-tuple of variables), each containing k bits.*
Verifier picks k clauses randomly and in each picks a variable randomly. This gives a k-tuple of clauses and a k-tuple of variables, and it reads the corresponding entries in the tables. It accepts if the $3k$ bits in T_1 constitute a satisfying assignment for the clause tuple and the assignment in T_2 to the variable tuple agrees with this assignment.

LEMMA 18.29 (RAZ)
There is constant c_{raz} such that for every $k \geq 1$ the completeness of V_{raz} is 1 and the soundness is c_{raz}^k.

Now we describe Håstad's verifier V_H. It is essentially V_{raz}, but it expects each entry in the two tables to be encoded using the long code. It expects these encodings to be *bifolded*, a property we now define (it is needed for technical reasons in Håstad's proof). It is motivated by the observation that coordinate functions satisfy $\chi_{\{w\}}(-x) = -\chi_{\{w\}}(x)$.

DEFINITION 18.13 *A function $f : \{-1,1\}^n \to \{-1,1\}$ is* bifolded *if for all $x \in \{-1,1\}^n$, $f(-x) = -f(x)$.*

Whenever the the membership proof is supposed to contain a longcode (i.e., a longcode encoding of a string), then we may assume without loss of generality that the function is bifolded. The reason is that the verifier can identify, for each pair of inputs $x, -x$, one designated representative —say x— and just define $f(-x)$ to be $-f(x)$. (One benefit — though of no consequence in the proof— of this convention is that bifolded functions require only half as many bits to represent.)

LEMMA 18.30
If $f : \{-1,1\}^n \to \{-1,1\}$ is bifolded and $\hat{f}_\alpha \neq 0$ then $|\alpha|$ must be an odd number (and in particular, nonzero).

PROOF: By definition,

$$\hat{f}_\alpha = \tfrac{1}{2^n} \sum_x f(x) \prod_{i \in \alpha} x_i.$$

If $|\alpha|$ is even then $\prod_{i\in\alpha} x_i = \prod_{i\in\alpha}(-x_i)$. So if f is bifolded, the terms corresponding to x and $-x$ have opposite signs and the entire sum is 0. □

Recall that Raz's verifier expects each entry of the first table to be a satisfying assignment to variables in a k-tuple of clauses; the number of such assignments for each tuple is 7^k, since for each clause of size 3, 7 out of $2^3 = 8$ assignments to the variables satisfy it. Each entry of the second table is an assignment to a k-tuple of variables; the number of these is 2^k. Let $W = 7^k$ and $U = 2^k$. Thus V_{raz} can think of each entry in the two tables as an integer from $[1..W]$ and $[1..U]$ respectively. Notice, after V_{raz} has read an entry in table 1 (a satisfying assignment to the variables in the k-tuple of clauses), then there is a unique entry in table 2 that will make it accept. Let $\pi : [1..W] \to [1..U]$ denote this function; namely, $\pi(w) = u$ if the assignment described in u is the same as the assignment to the

same variables in w. Then V_{raz} may be viewed as reading two integers w, u from the two tables, and checking if $\pi(w) = u$.

For a string $y \in \{-1,1\}^U$ we denote by $\overline{y} \in \{-1,1\}^W$ the string such that for every $w \in [1..W]$, $\overline{y}_w = y_{\pi(w)}$. In other words, for each $u \in U$, the bit y_u appears in all coordinates of $\overline{y}$ that are indexed by integers in $\pi^{-1}(u) \subseteq W$.

V_H expects the entries of the first table to be encoded using the (bifolded) long code for $[1..W]$ and the entries of the second table using the (bifolded) long code for $[1..U]$. The verifier emulates V_{raz} and picks out locations in the two tables, but then instead of reading the long codes in their entirety it does a very simple test that is reminiscent of the long code test.

Verifier V_H: *Pick a k-tuple of clauses C and a k-tuple of variables D just as V_{raz} would. Let $A : \{-1,1\}^W \to \{-1,1\}$ be the function (claimed long code) provided for C in table 1 and let $B : \{-1,1\}^U \to \{-1,1\}$ be the function provided for D in table 2. Randomly pick $x \in \{-1,1\}^W$, $y \in \{-1,1\}^U$. Pick a random set of coordinates by picking each coordinate with probability ρ. Let $z \in \{-1,1\}^W$ be a string that is -1 in these coordinates and 1 elsewhere. Accept iff $A(x)B(y) = A(\overline{y} \odot x \odot z)$.*

Suppose A, B are long codes of two integers w, u satisfying $\pi(w) = u$ (in other words, V_{raz} would have accepted the assignments represented by these integers). Then

$$A(x)B(y)A(\overline{y} \odot x \odot z) = x_w y_u (\overline{y} \odot x \odot z)_w \tag{18.33}$$

$$= x_w y_u (y_{\pi(w)} x_w z_w) \qquad = z_w. \tag{18.34}$$

Hence V_H accepts iff $z_w = 1$, which happens with probability $1 - \rho$. The next Lemma, whose proof appears a little later, gives a weak converse. It says that if V_H accepts A, B with probability significantly more than $1/2$, then the Fourier transforms of A, B must be correlated. To formalize this we define for $\alpha \subseteq [1..W]$,

$$\pi_2(\alpha) = \left\{ u \in [1..U] : \ \left| \pi^{-1}(u) \cap \alpha \right| \text{ is odd} \right\} \tag{18.35}$$

Notice in particular that for *every* $u \in \pi_2(\alpha)$ there is at least one $w \in \alpha$ such that $\pi(w) = u$.

In the next Lemma $\delta_{C,D}$ is allowed to be negative.

LEMMA 18.31
Let C, D, A, B be as in the description of V_H and suppose V_H has picked C, D. If the further probability that it accepts A, B is $1/2 + \delta_{(C,D)}$, then

$$\sum_{\alpha \subseteq [1..W], \alpha \neq \emptyset} \hat{A}_\alpha^2 \hat{B}_{\pi_2(\alpha)} (1-2\rho)^{|\alpha|} \geq 2\delta_{(C,D)}.$$

PROOF: By hypothesis, A, B are such that $E[A(x)A(x \odot \overline{y} \odot z)B(y)] \geq 2\delta_{C,D}$. Replace A, B by their Fourier expansions. Note that A, B have different domains and so the Fourier functions are different in the two expansions. We use $\chi_{\alpha,W}$ for Fourier functions on the domain $[1..W]$ and $\chi_{\alpha,U}$ for those on the domain $[1..U]$.

$$E_{x,y,z}[A(x)A(x \odot \overline{y} \odot z)B(y)] = E_{x,y,z}\left[(\sum_{\alpha} \hat{A}_\alpha \chi_{\alpha,W}(x))(\sum_{\beta} \hat{B}_\beta \chi_{\beta,U}(y))(\sum_{\gamma} \hat{A}_\gamma \chi_{\gamma,W}(x \odot \overline{y} \odot z))\right]$$
$$= \sum_{\alpha,\beta,\gamma} \hat{A}_\alpha \hat{B}_\beta \hat{A}_\gamma E_{x,y,z}\left[\chi_{\alpha,W}(x)\chi_{\beta,U}(y)\chi_{\gamma,W}(x)\chi_{\gamma,W}(\overline{y})\chi_{\gamma,W}(z)\right]$$

By orthonormality this simplifies to

$$= \sum_{\alpha,\beta} \hat{A}_\alpha^2 \hat{B}_\beta E_{y,z}[\chi_{\beta,U}(y)\chi_{\alpha,W}(\overline{y})\chi_{\alpha,W}(z)]$$
$$= \sum_{\alpha,\beta} \hat{A}_\alpha^2 \hat{B}_\beta (1-2\rho)^{|\alpha|} E_y[\chi_{\alpha,W}(\overline{y})\chi_{\beta,U}(y)] \quad (18.36)$$

since $\chi_{\alpha,W}(z) = (1-2\rho)^{|\alpha|}$, as noted in our analysis of the long code test. Now we have

$$E_y[\chi_{\alpha,W}(\overline{y})\chi_{\beta,U}(y)] = E_y[\prod_{w\in\alpha} \overline{y}_w \prod_{u\in\beta} y_u] \quad (18.37)$$
$$= E_Y[\prod_{w\in\alpha} y_{\pi(w)} \prod_{u\in\beta} y_u], \quad (18.38)$$

which is 1 if $\pi_2(\alpha) = \beta$ and 0 otherwise. Hence (18.36) simplifies to

$$\sum_{\alpha} \hat{A}_\alpha^2 \hat{B}_{\pi_2(\alpha)} (1-2\rho)^{|\alpha|}. \quad (18.39)$$

Finally we note that since the functions are assumed to be bifolded, the Fourier coefficients $\hat{A}_\emptyset$ and $\hat{B}_\emptyset$ are zero. Thus those terms can be dropped from the summation and the Lemma is proved. □

Now we prove a corollary of Lemma 18.31 which completes the proof of Håstad's 3-bit PCP Theorem since k is allowed to be any constant and c_{raz} is a fixed constant.

COROLLARY 18.32
If ρ, δ satisfy $\rho\delta^2 > c_{raz}^k$ then V_H has soundness less than $1/2 + (c_{raz}^k/\rho)^{1/2}$.

PROOF: Suppose V_H accepts some pair of tables (T_1, T_2) with probability at least $1/2 + \delta$. We give a probabilistic construction of a pair of tables (T_1', T_2') such that V_{raz} accepts (T_1', T_2') with probability at least $\rho\delta^2$, which is more than c_{raz}^k, and hence Lemma 18.29 implies that the input is in the language. Thus the soundness of V_H has been shown to be less than $1/2 + \delta$.

Now we describe the probabilistic construction. For each clause-tuple pick an assignment as follows. Suppose the function provided in the corresponding entry in T_1 is A. We randomly pick a subset of assignments for this clause-tuple by picking each $\alpha \subseteq [1..W]$ with probability $\hat{A}_\alpha^2$; this is a valid distribution since Parseval's identity implies $\sum_\alpha \hat{A}_\alpha^2 = 1$. We note that this subset of assignments is nonempty since bifolding implies $\hat{A}_\emptyset = 0$. (Thus

in fact $\sum_{\alpha \neq \emptyset} \hat{A}_\alpha^2 = 1$, and this condition $\alpha \neq \emptyset$ is also implicit elsewhere in this proof.) Finally, we pick a random element of α; this is the entry for the clause-tuple in table T_1'. We construct table T_2' by picking an assignment for each variable-tuple in the same way by sampling from the Fourier transform. Thus we have described a distribution on table pairs (T_1', T_2').

From now on, we reserve (C, D) to denote a clause k-tuple C and a variable k-tuple D that V_{raz} would pick. Let $I_{(C,D)}$ be a random variable (over the choice of T_1', T_2') such that

$$I_{(C,D)} = \begin{cases} 1 & \text{if values for (C, D) in tables } (T_1', T_2') \text{ satisfy } V_{raz} \\ 0 & \text{else} \end{cases} \tag{18.40}$$

Then $E_{(C,D)}[I_{(C,D)}]$ is the acceptance probability for V_{raz} using tables (T_1', T_2'). (Here $E_{(C,D)}$ denotes expectation over the distribution over (C, D) imposed by V_{raz}.)

We show that

$$E_{(T_1', T_2')}[E_{(C,D)}[I_{(C,D)}]] \geq \rho\delta^2, \tag{18.41}$$

whence it follows that there *exist* (T_1', T_2') which V_{raz} accepts with probability at least $\rho^2\delta$, and the Corollary will be proved.

To prove (18.41) we first notice that linearity of expectation allows us to exchange the order of the two expectations in (18.41): $E_{(T_1', T_2')}[E_{(C,D)}[I_{(C,D)}]] = E_{(C,D)}[E_{(T_1', T_2')}[I_{C,D}]]$. Consider any fixed (C, D) and let $1/2 + \delta_{(C,D)}$ be the probability that V_H accepts T_1, T_2 when it picks (C, D). Then by assumption on V_H

$$E_{(C,D)}[\delta_{(C,D)}] = \delta.$$

Suppose A, B were the long codes provided for C, D in tables T_1, T_2. What is the chance that the pair of assignments w, u picked for them in T_1', T_2' satisfy V_{raz} (namely, $u = \pi(w)$)? One sufficient (but not necessary) condition is that the sets of assignments α, β picked for them satisfy $\beta = \pi_2(\alpha)$ and that the assignments w, u picked from α, β respectively satisfy $u = \pi(w)$. The probability (over the choice of (T_1', T_2')) that this condition is satisfied is lowerbounded by

$$\sum_\alpha \frac{1}{|\alpha|} \hat{A}_\alpha^2 \hat{B}_{\pi_2(\alpha)}^2 \tag{18.42}$$

This is similar to but not quite the same as the expression in Lemma 18.31, according to which

$$2\delta_{(C,D)} \leq \sum_\alpha \hat{A}_\alpha^2 \hat{B}_{\pi_2(\alpha)} (1 - 2\rho)^{|\alpha|}. \tag{18.43}$$

However, since $(1 - 2\rho)^{|\alpha|} \leq \dfrac{2}{\sqrt{\rho\,|\alpha|}}$ (simple exercise) this is

$$\leq \sum_\alpha \hat{A}_\alpha^2 \left| \hat{B}_{\pi_2(\alpha)} \right| \frac{2}{\sqrt{\rho\,|\alpha|}} \tag{18.44}$$

Hence

$$\sum_\alpha \hat{A}_\alpha^2 \left| \hat{B}_{\pi_2(\alpha)} \right| \frac{1}{\sqrt{|\alpha|}} \geq \delta_{(C,D)} \sqrt{\rho} \tag{18.45}$$

Now we use Cauchy-Schwartz inequality, $\sum_i a_i b_i \leq (\sum_i a_i^2)^{1/2}(\sum_i b_i^2)^{1/2}$, with $\hat{A_\alpha}\left|\hat{B}_{\pi_2(\alpha)}\right|\frac{1}{\sqrt{|\alpha|}}$ playing the role of the a_i's and $\hat{A_\alpha}$'s playing that of the b_i's. We obtain

$$\sum_\alpha \hat{A}_\alpha^2 \left|\hat{B}_{\pi_2(\alpha)}\right| \tfrac{1}{\sqrt{|\alpha|}} \leq \left(\sum_\alpha \hat{A}_\alpha^2\right)^{1/2} \left(\sum_\alpha \hat{A_\alpha}^2 \hat{B}^2_{\pi_2(\alpha)} \tfrac{1}{|\alpha|}\right)^{1/2} \tag{18.46}$$

$$\leq \left(\sum_\alpha \hat{A_\alpha}^2 \hat{B}^2_{\pi_2(\alpha)} \tfrac{1}{|\alpha|}\right)^{1/2} \qquad \text{by Parseval} \tag{18.47}$$

This is for a particular choice of clause-tuple C and variable-tuple D. Then by (18.42) we have

$$E_{(C,D)}E_{(T_1',T_2')}[I_{C,D}] \geq E_{C,D}\left[\sum_\alpha \frac{1}{|\alpha|}\hat{A}_\alpha^2 \hat{B}^2_{\pi_2(\alpha)}\right] \tag{18.48}$$

since $E[X^2] \geq E[X]^2$ this is

$$\geq \left(E_{C,D}\left[\left(\sum_\alpha \frac{1}{|\alpha|}\hat{A}_\alpha^2 \hat{B}^2_{\pi_2(\alpha)}\right)^{1/2}\right]\right)^2 \tag{18.49}$$

$$\geq E_{(C,D)}[\delta^2_{(C,D)}\rho] \tag{18.50}$$

$$\geq \rho(E_{C,D}[\delta_{(C,D)}])^2 \qquad (\text{using } E[X^2] \geq E[X]^2 \text{ again}) \tag{18.51}$$

$$\geq \rho\delta^2, \tag{18.52}$$

and (18.41) has been proved. □

18.7 Other PCP Theorems: A survey

Other PCP-like classes have been defined by using variants of the definition above, and shown to equal NP (when the parameters are chosen appropriately). We mention some variants and the best results known for them.

1. The probability 1 in condition (**??**) may be allowed to be $c < 1$. Such a verifier is said to have *imperfect completeness* c.

2. The probability 1/2 in condition (**??**) may be allowed to be $s < c$. Such a verifier is said to have *soundness* s. Using standard results on random walks on expanders, it can be shown from the PCP theorem that every NP language has verifiers with perfect completeness that use $O(k)$ query bits for soundness 2^{-k} (here $k \leq O(\log n)$).

3. The number of query bits, which was $O(q(n))$ above, may be specified more precisely together with the leading constant. The constant is important for many inapproximability results. Building upon past results on PCPs and using fourier analysis,

Håstad [?] recently proved that for each $\epsilon > 0$, every NP language has a verifier with completeness $1 - \epsilon$, soundness $1/2$ and only 3 query bits. He uses this to show the inapproximability of MAX-3SAT upto a factor $8/7 - \epsilon$.

4. The *free bit* parameter may be used instead of query bits [?, ?]. This parameter is defined as follows. Suppose the query bit parameter is q. After the verifier has picked its random string, and picked a sequence of q addresses, there are 2^q possible sequences of bits that could be contained in those addresses. If the verifier accepts for only t of those sequences, then we say that the free bit parameter is $\log t$ (note that this number need not be an integer).

5. *Amortized free bits* may be used [?]. This parameter is defined as $\lim_{s \to 0} f_s / \log(1/s)$, where f_s is the number of free bits needed by the verifier to make soundness $< s$. Håstad has shown that for each $\epsilon > 0$, every NP language has a verifier that uses $O(\log n)$ random bits and ϵ amortized free bits [?]. He uses this to show (using a reduction from [?] and modified by [?, ?]) that MAX-CLIQUE is inapproximable upto a factor $n^{1-\epsilon}$.

6. The proof may contain not bits but letters from a larger alphabet Σ. The verifier's soundness may then depend upon Σ. In a *p prover 1-round interactive proof system*, the proof consists of p arrays of letters from Σ. The verifier is only allowed to query 1 letter from each array. Since each letter of Σ is represented by $\lceil \log |\Sigma| \rceil$ bits, the number of *bits* queried may be viewed as $p \cdot \lceil \log |\Sigma| \rceil$. Constructions of such proof systems for NP appeared in [?, ?, ?, ?, ?, ?]. Lund and Yannakakis [?] used these proof systems to prove inapproximability results for SETCOVER and many subgraph maximization problems. The best construction of such proof systems is due to Raz and Safra [?]. They show that for each $k \leq \sqrt{\log n}$, every NP language has a verifier that uses $O(\log n)$ random bits, has $\log |\Sigma| = O(k)$ and soundness 2^{-k}. The parameter p is $O(1)$.

Problems

1. Show that if SAT $\in$ PCP$(r(n), 1)$ for $r(n) = o(\log n)$ then P = NP. (Thus the PCP Theorem is probably optimal up to constant factors.)

2. Show that the PCP Theorem is equivalent to the statement that the reduction described in (18.2),(18.3) exists.

3. Show that a transitive graph is a union of vertex-disjoint cliques.

4. Let $q > (d+1)(\frac{q}{q-1})$. Show that if a function $f : F^m \to F$ where $F = GF(q)$ and its restriction on every line is a univariate degree d polynomial, then f is a degree d polynomial.

5. Suppose X, Y are i.i.d. random variables and $\Pr[X = Y] = 1 - \gamma$. Then show that there is a value a such that $\Pr[X = a] \geq 1 - \gamma$.

6. (*Well-distribution lemma for lines*) Let $S \subseteq F^m$ be a set whose size is $\mu \cdot q^m$. For every $K > 0$, at least $1 - \frac{1}{K^2}$ fraction of lines in F^m have between $\mu q(1 - \frac{K}{\sqrt{\mu q}})$ and $\mu q(1 + \frac{K}{\sqrt{\mu q}})$ points from S. (*Hint: A random line is picked by choosing a random pair of points in F^m. Now use almost-pairwise independence and Chebyshev.*)

7. (List decodability of polynomials) Let $f : F^m \to F$ be any function. Suppose integer $d \geq 0$ and fraction ρ satisfy $\rho > 2\sqrt{\frac{d}{q}}$. Then there are at most $2/\rho$ degree d polynomials that have agreement at least ρ with f. (*Hint: The first polynomial describes f in a ρ fraction of points say S_1, the second polynomial describes f in $\rho - d/q$ fraction of points S_2 where $S_1 \cap S_2 = \emptyset$, etc.*)

8. These little problems introduce you to the reasoning used in the analysis of the low degree test. In all of them assume that the field size q is at least quadratic in the degree in question.

   (a) Show that if $f : F^m \to F$ is described by a univariate polynomial of degree $\leq d$ on all lines in F^m, then f is a polynomial of degree at most d.

   (b) Suppose $f : F^2 \to F$ and for each $a \in F$ let C_a, R_a be two univariate degree d polynomials. Suppose these satisfy for at least $(1 - \frac{1}{20d})$ fraction of $(a, b) \in F^2$,

   $$f(a, b) = R_a(b) = C_b(a).$$

   Then show that f is 0.2-close to a bivariate polynomial whose degree in each variable is at most d.

   (c) Show that if $f, g : F^m \to F$ are δ-close (i.e., agree on δ fraction of points) then on at least $1 - 1/K^2$ fraction of lines in F^m, functions f, g agree on between $\delta(1 - \frac{2K}{\sqrt{\delta q}})$ and $\delta(1 + \frac{2K}{\sqrt{\delta q}})$ fraction of points on the line. (Hint: A random line is picked by picking two random points, so the set of points on it are roughly pairwise independent. Check this by computing the variance!)

9. (A simple PCP Theorem using logspace verifiers) Using the fact that a correct tableau can be verified in logspace, we saw the following exact characterization of NP:

   $$\text{NP} = \{L : \text{ there is a logspace machine } M \text{ s.t } x \in L \text{ iff } \exists y \ : \ M \text{ accepts } (x, y).\}.$$

   Note that M has two-way access to y.

   Let L-PCP($r(n)$) be the class of languages whose membership proofs can be probabilistically checked by a logspace machine that uses $O(r(n))$ random bits but makes only one pass over the proof. (To use the terminology from above, it has 2-way access to x but 1-way access to y.) As in the PCP setting, "probabilistic checking of membership proofs" means that for $x \in L$ there is a proof y that the machine accepts with probability 1 and if not, the machine rejects with probability at least 1/2. In this problem we prove NP = L-PCP($\log n$).

   Assume the following fact (which is only asymptotically correct) from now on: The number of primes in the interval $[l, 2l]$ is exactly $\frac{l}{\log l}$.

(a) Show that L-PCP($\log n$) $\subseteq$ NP.

(b) Show that if x and y are distinct n-bit numbers, and p is a prime chosen randomly from among the first $2n$ primes, then

$$Pr[x = y \bmod p] \leq \frac{1}{2}.$$

(c) Let S_1 and S_2 be two given multisets with at most m elements, each of which is an n-bit numbers. Describe a probabilistic algorithm that tests whether or not S_1, S_2 are equal using $O(\log n + \log m)$ random bits and using only one pass over S_1, S_2. If $S_1 = S_2$ the algorithm must accept with probability 1 and if $S_1 \neq S_2$ it must reject with probability at least $1 - 1/n$. (Hint: Interpret a multiset as a multivariate polynomial and use Schwartz-Zippel.) In order to be useful below, your algorithm should work even if the elements of S_1, S_2 are arbitrarily interspersed.

(d) Let ψ be a $3SAT$ formula. This part concerns a special representation of a satisfying assignment. In the rest of the question, "pair" refers to some (n, v) where n is a variable in ψ and $v \in \{0, 1\}$.

A *triple* for a clause in ψ is a tuple of the type:

$$< (n_1, v_1), (n_2, v_2), (n_3, v_3) >$$

where n_1, n_2 and n_3 are variables occuring in the clause. The triple is *valid* if the assignment v_1, v_2 and v_3 to these variables makes the clause true.

A "valuation sequence" σ for ψ consists of two parts:

- A sequence of pairs (n, v) where n is a variable of ψ. Call this sequence the *assignment part* of σ.
- A sequence of valid triples, each representing some clause of ψ. Call this part the *evaluation part* of σ.

σ is *complete* if there each variable is assigned exactly once in its assignment part, and each clause is evaluated exactly once in its evaluation part.

It is *globally consistent* if for each variable n, every pair (n, v) appearing in σ has the same value of v. (Intuition: A pair (n, v) occuring anywhere implies that variable n has been assigned value v, and consistency demands that this value is the same throughout σ).

Convince yourself that φ is satisfiable iff it has a complete and globally consistent valuation sequence.

Now complete the proof of NP $\subseteq$ L-PCP($r(n)$).

(e) Suppose we define $J-PCP(r(n))$ similarly to $L-PCP(r(n))$, except the verifier is only allowed to read $O(r(n))$ successive bits in the membership proof. (It can decide which bits to read.) Then show that $J - PCP(\log n) \subseteq$ L.

10. If $s(n)q(n) \leq n$ and $r(n) \geq \log n$ then show that PCP($r(n), q(n)$) $\subseteq$ sPCP($r(n), 1, \text{poly}(q(n))$). (Hint: Change the PCP verifier so that it expects the proof string to be a polynomial extension of the original proof. Then use the low degree test and the reconstruction procedure.)

11. (*Stronger form of Low Degree Test*) The goal of the next few questions is to prove a stronger form of the Low Degree Test.