

# CMPT 710 - Complexity Theory: Lecture 24

Valentine Kabanets

November 29, 2007

## 1 PCP Theorem: Fragments of the proof

To talk about PCPs, we need the definition of a probabilistic polytime verifier  $V$  that is  $(r(n), q(n))$ -restricted, i.e., on input of size  $n$ ,

1.  $V$  uses at most  $O(r(n))$  random bits, and
2.  $V$  queries at most  $O(q(n))$  bits of the proof string.

We say that  $L \in \text{PCP}[r(n), q(n)]$  if there is a probabilistic polytime  $(r(n), q(n))$ -restricted verifier  $V$  such that

1. (**completeness**)  $x \in L \Rightarrow \exists \pi_x \Pr[V^{\pi_x}(x) = 1] = 1$ ,
2. (**soundness**)  $x \notin L \Rightarrow \forall \pi \Pr[V^{\pi}(x) = 1] \leq 1/2$ .

Here, the probabilities are taken over the random choices of the verifier, and the notation  $V^{\pi}$  means that the verifier  $V$  has random access to the string  $\pi$ , i.e.,  $V$  may request to see the  $i$ th bit of the string  $\pi$ , for any  $i$ .

A verifier  $V$  can be either *non-adaptive* or *adaptive*. A non-adaptive verifier, after flipping its  $O(r(n))$  random coins, decides upon  $O(q(n))$  locations in the proof string that  $V$  wants to see. In other words, a non-adaptive verifier decides upon what locations to see in advance.

On the other hand, an adaptive verifier, after flipping  $O(r(n))$  random coins, decides upon a first location of the proof string that  $V$  wants to see. Then, after seeing the contents of this location,  $V$  decides which location to see next. After seeing the contents of the second location,  $V$  decides which location to see third, and so on for  $O(q(n))$  locations.

In general, when talking about PCP verifiers, we mean adaptive verifiers. However, most verifiers we'll see will be non-adaptive.

A couple of remarks about the definition of PCP. First, note that a non-adaptive  $(r(n), q(n))$ -restricted verifier can access (over all of its random string of length  $O(r(n))$ ) at most  $2^{O(r(n))}O(q(n))$  bits of the proof string. So, if  $r(n) = \log n$ , and  $q(n) \in \text{poly}(n)$ , then such a verifier can see at most a polynomial number of bits of the proof, and so we may as well assume that the proof string is of polynomial length (since if it's longer, the verifier won't be able to see the extra part anyway).

The case of an adaptive verifier is trickier. Once  $V$  has flipped its random coins, we can think of the *strategy* of  $V$  for requesting the bits of the proof string as a binary tree of height

at most  $O(q(n))$ . The root  $u$  of the tree is labeled by the first location of the proof string that  $V$  wants to see. The left child  $v$  of  $u$  is labeled by the next location in the proof that  $V$  wants to see if the contents of the first location is 0; the right child  $w$  of  $u$  is labeled by the next location in the proof that  $V$  wants to see if the contents of the first location is 1. The children of  $v$  and  $w$  are defined similarly. Since  $V$  asks at most  $O(q(n))$  queries, the height of this strategy tree is  $O(q(n))$ . Finally, the children of the last query in a given path of the strategy tree are two leaves: left (corresponding to the contents 0) and right (corresponding to the contents 1). The leaves are labeled by Accept or Reject, according to the decision of the verifier  $V$ .

For example, if  $q(n) = 2$ , then for some random string of  $V$ , the strategy tree of  $V$  may look as follows: root  $u$  is labeled 13, the left child  $v$  is labeled 5, and the right child  $w$  is labeled 7. Both children of  $v$  are labeled Accept; and the left child of  $w$  is labeled Accept, whereas the right child of  $w$  is labeled Reject. This tree corresponds to the strategy: if proof bit 13 is 0, then see proof bit 5; otherwise, see proof bit 7. If bit 13 is 1, and bit 7 is 1, then Reject; in all other cases, Accept.

We now state the famous PCP theorem, which is one of the deepest results in theoretical computer science in the last decade.

**Theorem 1** (PCP Theorem).  $\text{NP} = \text{PCP}[\log n, 1]$ .

The PCP Theorem was proved by Arora, Safra, and Arora, Lund, Motwani, Sudan, Szegedy in two papers that in turn built upon rich body of work on interactive proof systems done earlier by many researchers. Recently, the authors of these two papers have been honored with the Gödel prize for their contribution to computer science.

A complete proof of the PCP Theorem is beyond this introductory course. Instead, we'll look at the following weaker version of the PCP Theorem, which is already quite non-trivial, and furthermore, this weaker version is used as a building block in the proof of the general PCP Theorem.

**Theorem 2** (Weak PCP Theorem).  $\text{NP} = \text{PCP}[\log n, \text{poly log } n]$

The inclusion  $\text{PCP}[\log n, \text{poly log } n] \subseteq \text{NP}$  is easy. The interesting direction is  $\text{NP} \subseteq \text{PCP}[\log n, \text{poly log } n]$ .

The rest of these notes is devoted to the proof of the Weak PCP Theorem stated above.

## 2 Arithmetization of 3SAT

A 3-CNF formula  $\phi(x_1, \dots, x_n) = C_1 \wedge \dots \wedge C_t$ , where each clause  $C$  is of the form  $[x_{i_1} = b_1 \vee x_{i_2} = b_2 \vee x_{i_3} = b_3]$  where  $b_1, b_2, b_3 \in \{0, 1\}$ . (In this notation, a clause  $(\bar{x} \vee y \vee z)$  corresponds to  $[x = 0 \vee y = 1 \vee z = 1]$ .)

It will be convenient for us to view  $\phi$  as an *indicator function*

$$\phi : \{1, 2, \dots, n\}^3 \times \{0, 1\}^3 \rightarrow \{0, 1\},$$

so that  $\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$  iff the clause  $[x_{i_1} = b_1 \vee x_{i_2} = b_2 \vee x_{i_3} = b_3]$  is present in the formula  $\phi$ .

Let  $S \subseteq \mathbb{F}$  be a subset of some finite field  $\mathbb{F}$ . A standard way to turn a function  $g : S \rightarrow \{0, 1\}$  into a polynomial over  $\mathbb{F}$  is as follows.

For each  $a \in S$ , define a polynomial  $l_a(x)$  such that  $l_a(a) = 1$  and, for every  $b \in S \setminus \{a\}$ ,  $l_a(b) = 0$ . By standard interpolation, such a polynomial can be constructed from the values at  $|S|$  distinct points, and the resulting polynomial is of degree at most  $|S|$ .

Now, define the polynomial extension of  $g$  to  $\mathbb{F}$  as

$$\hat{g}(x) = \sum_{a \in S} g(a) * l_a(x).$$

Note that  $\hat{g}(x) = g(x)$  for each  $x \in S$ . Also, by what we argued earlier about the degree of each  $l_a$ , we get that the degree of  $\hat{g}(x)$  is at most  $|S|$ .

This way, any Boolean-valued function defined on a set of size  $n$  can be extended to a polynomial of degree at most  $n$ , so that the new polynomial agrees with the old function over the domain of the old function (of size  $n$ ).

Now, in our case, we could extend the indicator function  $\phi$  to a polynomial over some finite field of size  $> n$  so that the degree of the resulting polynomial is about  $n$ . However, for our purposes of proving the PCP Theorem, the degree  $n$  is way too much. We need degree  $\text{poly log } n$  instead!

Therefore, we need to use a different way of encoding a function  $g : S \rightarrow \{0, 1\}$  for a set  $S$  of size  $n$ . The idea is to use *multivariate* polynomials instead of univariate ones. Pick a subset  $H$  of size  $h = \text{poly log } n$  and  $m = O(\log n / \log \log n)$  so that  $h^m = n$ . Identify the elements  $1, 2, \dots, n$  with the tuples of  $H^m$  in some canonical way. Now, view the function  $g$  as a function  $g : H^m \rightarrow \{0, 1\}$ .

Define for each  $a \in H$ , the polynomial  $l_a(x)$  of degree at most  $|H|$  such that  $l_a(a) = 1$  and  $l_a(b) = 0$  for each  $b \in H \setminus \{a\}$ . We extend  $g$  to a polynomial  $\hat{g}$  as follows:

$$\hat{g}(x_1, \dots, x_m) = \sum_{(a_1, \dots, a_m) \in H^m} g(a_1, \dots, a_m) * l_{a_1}(x_1) * \dots * l_{a_m}(x_m).$$

It is easy to check that  $\hat{g}(x) = g(x)$  for every element  $x \in H^m$ , and the degree of  $\hat{g}$  in each variable is at most  $|H|$  (which is the degree of each of  $l_a$ ).

Note that, as a result of this multivariate encoding, we get a polynomial of degree  $\text{poly log } n$  in each of  $O(\log n)$  variables, and so of total degree  $\text{poly log } n$ , which is what we wanted.

Thus, to encode  $\phi : \{1, \dots, n\}^3 \times \{0, 1\}^3 \rightarrow \{0, 1\}$ , we choose  $H$  of size  $h$ , and  $m$  as specified above (so that  $h^m = n$ ). We view  $\phi$  as a function from  $H^{3m} \times \{0, 1\}^3$  to  $\{0, 1\}$ , and extend  $\phi$  to a  $(3m + 3)$ -variate polynomial  $f$  of degree at most  $h$  in each variable over a finite field  $\mathbb{F}$  of size  $\text{poly log } n$ , in the way described above. So, the polynomial  $f$  agrees with  $\phi$  over the domain of  $\phi$ , and  $f$  has total degree  $\text{poly log } n$ , which by our choice of the size of  $\mathbb{F}$  can be made much smaller than the size of  $\mathbb{F}$ .

Now, we have that  $\phi \in \text{SAT}$  iff there is an assignment function  $a : H^m \rightarrow \{0, 1\}$  such that, for every  $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in H^{3m} \times \{0, 1\}^3$ ,

$$\phi(z) = 0 \vee a(i_1) = b_1 \vee a(i_2) = b_2 \vee a(i_3) = b_3.$$

The interpretation is that  $a$  encodes an assignment to  $n$  variables of  $\phi$  (since  $1, 2, \dots, n$  can be identified with the set  $H^m$  of size  $n$ ) such that, for each clause present in  $\phi$ ,  $a$  satisfies that clause.

Let  $A : \mathbb{F}^m \rightarrow \mathbb{F}$  be a  $(\text{poly log } n)$ -degree polynomial extension of the function  $a$  to the entire finite field  $\mathbb{F}$ . Let  $f$  be the polynomial extension of  $\phi$  described earlier. Then the condition above can be restated as follows: for every  $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in H^{3m} \times \{0, 1\}^3$ ,

$$f(z)(A(i_1) - b_1)(A(i_2) - b_2)(A(i_3) - b_3) = 0. \quad (1)$$

The following claim is easy to prove.

**Claim 3.**  $\phi \in \text{SAT}$  iff there exists an  $m$ -variate polynomial  $A : \mathbb{F}^m \rightarrow \mathbb{F}$  of degree at most  $h$  in each variable such that for every  $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in H^{3m} \times \{0, 1\}^3$ , equality (1) holds.

So, our PCP Verifier will expect to be given random access to a low-degree polynomial  $A : \mathbb{F}^m \rightarrow \mathbb{F}$ , and will try to check equality (1) for all  $z \in H^{3m} \times \{0, 1\}^3$ .

**Two problems.** (1) How does a verifier make sure that the function  $A$  is indeed a low-degree polynomial? Note that the verifier has limited access to the proof, i.e., can read at most  $\text{poly log } n$  bits of the proof. On the other hand, the description of a function  $A$  is polynomial in  $n$ . (2) How does a verifier check equality (1) for all  $z \in H^{3m} \times \{0, 1\}^3$ ? Note that the number of such  $z$ 's is at least  $n^3$ , and the verifier can read at most  $\text{poly log } n$  bits of the PCP proof.

The first problem is circumvented by two steps. First, the verifier probabilistically checks that  $A$  is sufficiently close to some low-degree polynomial  $\tilde{A}$ . Such low-degree testing algorithms exist, but their analysis is beyond these lecture notes. Secondly, after verifying that  $A$  is close to some low-degree polynomial  $\tilde{A}$ , the verifier can run another randomized procedure that, given oracle access to  $A$ , will compute  $\tilde{A}(x)$  on every  $x$  with high probability. This is what Problem Set 4, problem 2, asks you to show.

So, doing these two steps (low-degree test and error-correction), we may assume that the verifier has access to a real low-degree polynomial  $A$ .

The second problem is tackled in the next section.

### 3 Making constraints robust

Since both  $f$  and  $A$  are low-degree polynomials, the left-hand side of expression (1) is also a low degree polynomial. Let us denote this polynomial by  $\tilde{P}_0(z)$ . The total degree of  $\tilde{P}_0$  is at most  $2h(3m + 3) = \text{poly log } n$ .

Let  $P_0(z)$  be a polynomial contained in the PCP proof (in addition to the assignment polynomial  $A$ ), so that  $P_0$  is supposedly equal to the polynomial  $\tilde{P}_0$ . First, the verifier will run low-degree testing and error correction on  $P_0$ . Then the verifier will check that  $P_0 \equiv \tilde{P}_0$ , by testing the equality of these two polynomials on a random point.

The inputs  $z$  come from the set  $H^{3m} \times \{0, 1\}$ , so that each of the first  $3m$  variables take one of the  $h$  possible values. We would like to use the Schwartz-Zippel lemma to argue that

if  $P_0 \neq 0$ , then  $P_0(z) \neq 0$  for a large fraction of  $z$ 's. However, applying the Schwartz-Zippel to  $P_0(z)$  only gives us that the probability that a non-zero polynomial  $P_0(z)$  is zero for a random point in  $H^{3m+3}$  is at most the degree of the polynomial (which is  $2h(3m+3)$ ) divided by the size of the sample domain for each of the  $(3m+3)$  variables (which is  $h$ ). So, the Schwartz-Zippel only gives the probability at most  $2(3m+3) > 1$ , which is completely useless.

The problem is that equality (1) is about the polynomial  $P_0$  whose total degree is not sufficiently smaller than the size of  $H$  (the sample domain for each of the variables). We would like to be able to extend the equality to a larger domain for each of the variables. It turns out that this is indeed possible to do.

First, let us look at a simple example. Let  $H = \{b_1, \dots, b_h\}$  be a subset of a field  $\mathbb{F}$ . Given a polynomial  $p(x)$  over  $\mathbb{F}$ , define a new polynomial

$$q(y) = \sum_{j=1}^h p(b_j)y^j.$$

Observe that  $p = 0$  on each element of  $H$  iff  $q \equiv 0$  over the entire field  $\mathbb{F}$ .

Using this idea, we can construct a sequence of polynomials  $P_1, \dots, P_l$ , for  $l = 3m+3$ , by defining

$$P_i(y_1, \dots, y_i, x_{i+1}, \dots, x_l) = \sum_{j=1}^h P_{i-1}(y_1, \dots, y_{i-1}, b_j, x_{i+1}, \dots, x_l)y_i^j.$$

It is not difficult to verify that  $P_i = 0$  over the domain  $\mathbb{F}^i \times H^{l-i}$  iff  $P_{i-1} = 0$  over the domain  $\mathbb{F}^{i-1} \times H^{l-i+1}$ . (Here, for the sake of simplicity, we changed the last three argument of  $P_0$  to be from the domain  $H^3$  rather than  $\{0, 1\}^3$ . In general, one just treats these last three arguments differently from the rest.)

So, for the last polynomial  $P_l$ , we have that  $P_l \equiv 0$  over the entire domain  $\mathbb{F}^l$  iff  $P_0 = 0$  over the domain  $H^l$ .

## 4 Description of the Verifier

The PCP proof will need to contain the information about a (supposedly) satisfying assignment  $A$ , descriptions of polynomials  $P_0, P_1, \dots, P_l$  (which supposedly come from the process described in the previous section), plus the information needed for the low-degree testing algorithm to verify that  $A, P_0, P_1, \dots, P_l$  are all sufficiently close to some  $(\text{poly log } n)$ -degree polynomials.

Given oracle access to the PCP proof, the verifier will first run all low degree tests. If one of them fails, the verifier will Reject. Suppose all the tests pass. Whenever the verifier needs the value of some polynomial (say,  $\tilde{A}(x)$ , for the low-degree polynomial  $\tilde{A}$  that is sufficiently close to the function  $A$  presented in the PCP proof), it will run the probabilistic error-correction procedure (on  $A$ , in order to compute  $\tilde{A}(x)$ ). So, for the rest of the section, let's assume that  $A, P_0, \dots, P_l$  are all low-degree polynomials.

The verifier has to check that, for every  $z = (z_1, \dots, z_l) \in \mathbb{F}^l$ ,

1. (C0):  $P_0(z) = \tilde{P}_0(z)$ , where  $\tilde{P}_0$  is the polynomial on the left-hand side of equation (1),
2. for each  $i = 1, 2, \dots, l$ , (Ci):

$$P_i(z_1, \dots, z_i, z_{i+1}, \dots, z_l) = \sum_{j=1}^h P_{i-1}(z_1, \dots, z_{i-1}, h_j, z_{i+1}, \dots, z_l) z_i^j,$$

3. (C(l + 1)):  $P_l(z) = 0$ .

Note that conditions  $C1, \dots, Cl$  ensure that the sequence  $P_0, \dots, P_l$  is indeed the sequence obtained from  $P_0$  as described in the previous section.

Observe that each of the conditions  $C0, \dots, C(l + 1)$  is a polynomial identity testing for polynomials of degree  $d + h$ , where  $d = (3m + 3)h$  is (upper bound on) the degree of  $P_i$ ,  $0 \leq i \leq l$ . Since the field size  $|\mathbb{F}| = \text{poly log } n$  can be chosen sufficiently larger than  $d + h$ , we get the following.

**Lemma 4.** *If  $P_0, \dots, P_l, \tilde{P}_0$  are polynomials of degree at most  $d$ , then for the collection of constraints*

$$\{C0(z) \wedge C1(z) \wedge \dots \wedge C(l + 1)(z)\}_{z \in \mathbb{F}^l},$$

*we have: either all of the constraints are satisfied, or at most the fraction  $\frac{(d+h)}{|\mathbb{F}|} = o(1)$  of them are satisfied.*

Thus, if our verifier checks the constraint

$$C0(z) \wedge C1(z) \wedge \dots \wedge C(l + 1)(z)$$

for a random  $z \in \mathbb{F}^l$ , accepting iff that constraint is satisfied, then, with probability  $1 - o(1)$ , the verifier rejects a wrong PCP proof. Of course, if the PCP proof is correct, then the verifier will accept with probability 1.

Since  $|\mathbb{F}^l| = \text{poly}(n)$ , to sample a random point  $z \in \mathbb{F}^l$ , the verifier needs only  $O(\log n)$  random bits. (The same requirements on the amount of randomness are true for low degree testing and error-correction algorithms performed by the verifier at the beginning.)

For a given point  $z \in \mathbb{F}^l$ , the verifier needs access the PCP proof to get  $\text{poly}(l) = \text{poly log } n$  field elements of size  $O(\log \log n)$ , for the total of  $\text{poly log } n$  bits of the proof. (The same  $\text{poly log } n$  requirement is also true for low-degree testing and error-correction algorithms performed by the verifier.)

Thus, our verifier is indeed a  $(O(\log n), \text{poly log } n)$ -restricted verifier for the language 3SAT. Hence,  $\text{NP} \subseteq \text{PCP}[\log n, \text{poly log } n]$ , as promised.

**Remark** The proof of the PCP Theorem (its weak version) outlined above follows the original arguments in the paper by Arora et al. Recently, Irit Dinur (STOC'06) discovered a beautiful alternative proof of the PCP Theorem, using expander graphs instead of arithmetization and low-degree polynomials. Thinking of PCP as a special kind of encoding, Dinur's proof essentially shows how to take PCP codes of constant size and transform them (iteratively) into PCP codes of a given size  $n$ . (This approach is very similar to that used by Reingold, Vadhan, and Wigderson to construct arbitrary-size expander graphs from constant-size expanders, using special graph operations — another recent breakthrough in complexity theory.)

## 5 Some topics that we missed

Here is an incomplete list of topics in complexity that we should have talked about in the course, but didn't have enough time for.

1. Randomized logspace computation, the complexity class  $RL$ , and “ $RL$  vs.  $L$ ” question.
2. The complexity class  $SL$ , which has Undirected Graph Reachability as a complete problem; the latter problem was recently shown to be in  $L$  by Omer Reingold (STOC'05).
3. Counting complexity class  $\#P$ , and its complete problems (counting the number of satisfying assignments in a 3cnf, and computing the Permanent of a Boolean matrix).
4. Relativization and “natural proofs” — meta-results that rule out certain approaches to the “ $P$  vs.  $NP$ ” problem.
5. Zero-knowledge interactive proof systems (in particular, for Graph Isomorphism and other problems in  $NP$ ).