PCP

- PCP[r(n),q(n)]: set of languages L with p.p.t. verifier V that has (r, q)-restricted access to a string “proof”
  - V tosses O(r(n)) coins
  - V accesses proof in O(q(n)) locations
  - (completeness) $x \in L \Rightarrow \exists$ proof such that $\Pr[V(x, proof) accepts] = 1$
  - (soundness) $x \notin L \Rightarrow \forall$ proof* $\Pr[V(x, proof*) accepts] \leq \frac{1}{2}$

Two observations:
- PCP[1, poly n] = NP
- PCP[log n, 1] ⊆ NP

The PCP Theorem (AS, ALMSS):
PCP[log n, 1] = NP.

The inner verifier

Theorem: NP ⊆ PCP[n^2, 1]

Proof (first steps):
1. Quadratic Equations is NP-hard
2. PCP for QE:
   - proof = all quadratic functions of a soln. x
   - verification = check that a random linear combination of equations is satisfied by x
   - (if prover keeps promise to supply all quadratic fns of x)

Quadratic Equations

- quadratic equation over $F_2$:
  $$\sum_{i<j} a_{i,j} X_i X_j + \sum_i b_i X_i + c = 0$$
- language QUADRATIC EQUATIONS (QE) = { systems of quadratic equations over $F_2$ that have a solution (assignment to the X variables) }
Quadratic Equations

Lemma: QE is NP-complete.

Proof: clearly in NP; reduce from CIRCUIT SAT
– circuit C an instance of CIRCUIT SAT
– QE variables = variables + gate variables

PCP for QE

If prover keeps promise to supply all quadratic fns of x, a solution of QE instance...
• Verifier’s action:
  – query a random linear combination R of the equations of the QE instance
  – Completeness: obvious
  – Soundness: x fails to satisfy some equation; imagine picking coeff. for this one last
    \[ Pr[x satisfies R] = 1/2 \]

PCP for QE

\[ f(x) = \sum a_i x_i \quad \text{Had}(x) \]
\[ g(x) = \sum_{i,j} A[i,j] x_i x_j \quad \text{Had}(x \otimes x) \]

• Linearity test: given access to \( h : \mathbb{F}^n \to \mathbb{F} \)
  – pick random \( a,b \); check if \( h(a) + h(b) = h(a+b) \); repeat \( O(1) \) times
  – do this for functions \( f \) and \( g \) supplied by prover

Theorem [BLR]: h linear \( \Rightarrow \) prob. success = 1; prob. success \( \geq 1 - \delta \) \( \Rightarrow \) exists linear \( h' \) s.t.
\[ Pr_u [ h'(a) = h(a) ] \geq 1 - O(\delta) \]

PCP for QE

\[ f(x) = \sum a_i x_i \quad \text{Had}(x) \]
\[ g(x) = \sum_{i,j} A[i,j] x_i x_j \quad \text{Had}(x \otimes x) \]

• Self-correction:
  – given access to \( h : \mathbb{F}^m \to \mathbb{F} \) close to linear \( h' \); i.e.,
    \[ Pr_u [ h'(a) = h(a) ] \geq 1 - O(\delta) \]
  – to access \( h'(a) \), pick random \( b \); compute
    \( h(b) + h(a+b) \)
  – with prob. at least \( 1 - 2O(\delta) \), \( h(b) = h'(b) \) and \( h(a+b) = h'(a+b) \); hence we compute \( h'(a) \)

Quadratic Functions Code

• intended proof:
  – \( F \) the field with 2 elements
  – given \( x \in F^n \), a solution to instance of QE
  – \( f : F^n \to F_2 \) all linear functions of \( x \)
    \[ f(a) = \sum a_i x_i \]
  – \( g : F^n \to F_2 \) includes all quadratic fns of \( x \)
    \[ g(A) = \sum_{i,j} A[i,j] x_i x_j \]
  – KEY: can evaluate any quadratic function of \( x \) with a single evaluation of \( f \) and \( g \)
### PCP for QE

- **Consistency check:** given access to linear functions \( f = \text{Had}(u) \) and \( g' = \text{Had}(v) \)
  - pick random \( a, b \in F^n \); check that
  - completeness: if \( V = u \otimes u \)
  - soundness: claim that if \( V = u \otimes u \)

\[
\Pr[(\sum u_i, \Sigma v_i) = \Sigma (a_i b_i V[i,j]) = g'(ab^t)]
\]

### NP \subseteq PCP[log n, polylog n]

- MAX-k-SAT
  - given: \( k\)-CNF \( \phi \)
  - output: max. # of simultaneously satisfiable clauses

- generalization: MAX-k-CSP
  - given:
    - variables \( x_1, x_2, ..., x_n \) taking values from field \( F_q \)
    - \( n = q^m \) for some integer \( m \)
    - \( k \)-ary constraints \( C_1, C_2, ..., C_t \)
  - output: max. # of simultaneously satisfiable constraints

### NP \subseteq PCP[log n, polylog n]

- **algebraic version:** MAX-k-PCS
  - given:
    - variables \( x_1, x_2, ..., x_n \) taking values from field \( F_q \)
    - \( n = q^m \) for some integer \( m \)
    - \( k \)-ary constraints \( C_1, C_2, ..., C_t \)
  - assignment viewed as \( f(F_q)^m \rightarrow F_q \)
  - output: max. # of constraints simultaneously satisfiable by an assignment that has deg. \( \leq d \)
**Lemma:** for every constant $1 > \epsilon > 0$, the 
MAX-$k$-PCS gap problem with 
- $t = \text{poly}(n)$ $k$-ary constraints with $k = \text{polylog}(n)$ 
- field size $q = \text{polylog}(n)$ 
- $n = q^m$ variables with $m = O(\log n / \log \log n)$ 
- degree of assignments $d = \text{polylog}(n)$ 
- gap $\epsilon$

is NP-hard.

**Proof of Lemma**
- reduce from 3-SAT
- 3-CNF $\varphi(x_1, x_2, \ldots, x_n)$
- can encode as $\psi':[n] \times [n] \times \{0,1\}^3 \rightarrow \{0,1\}$
  - $\psi'(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ iff $\varphi$ contains clause
    $(x_{i_1}^{b_1} \lor x_{i_2}^{b_2} \lor x_{i_3}^{b_3})$
  - e.g. $(x_3 \lor \neg x_1 \lor x_2) \Rightarrow \psi'(3,5,2,1,0,1) = 1$

**NP \subseteq PCP[\log n, \text{polylog } n]**

- pick $H \subseteq F_q$ with $0,1 \subseteq H$, $|H| = \text{polylog } n$
- pick $m = O(\log n / \log \log n)$ so $|H|^m = n$
- identify $[n]$ with $H^m$
- $\psi: H^m \times H^m \times H^m \times H^3 \rightarrow \{0,1\}$ encodes $\varphi$
- assignment $a: H^m \rightarrow \{0,1\}$
- Key: $a$ satisfies $\varphi$ iff $\forall i_1 i_2 i_3 b_1 b_2 b_3$
  - $\psi'(i_1, i_2, i_3, b_1, b_2, b_3) = 0$ or $a(i_1) = b_1$ or $a(i_2) = b_2$ or $a(i_3) = b_3$

**NP \subseteq PCP[\log n, \text{polylog } n]**

- $\psi': (F_q)^{3m+3} \rightarrow F_q$ encodes $\varphi$
  - $a': (F_q)^m \rightarrow F_q$ s.a. iff $\forall (i_1 i_2 i_3 b_1 b_2 b_3) \in H^{3m+3}$
  - $\psi'(i_1, i_2, i_3, b_1, b_2, b_3) = 0$ or $a'(i_1) = b_1$ or $a'(i_2) = b_2$ or $a'(i_3) = b_3$

  - define: $p_a: (F_q)^{3m+3} \rightarrow F_q$ from $a'$ as follows
    - $p_a(i_1 i_2 i_3 b_1 b_2 b_3) = \psi'(i_1 i_2 i_3 b_1 b_2 b_3)(a'(i_1) - b_1)(a'(i_2) - b_2)(a'(i_3) - b_3)$
  - $a'$ s.a. iff $\forall (i_1 i_2 i_3 b_1 b_2 b_3) \in H^{3m+3}$
  - $p_a(i_1 i_2 i_3 b_1 b_2 b_3) = 0$
NP \subseteq \text{PCP}[\log n, \text{polylog } n]

\psi' : (F_q)^{3m+3} \rightarrow F_q \text{ encodes } \varphi

a' : (F_q)^m \rightarrow F_q \text{ s.a. if } \forall (i_1, i_2, i_3, b_1, b_2, b_3) \in H^{3m+3}

p_{a}(i_1, i_2, i_3, b_1, b_2, b_3) = 0

– note: deg(p_a) \leq 2(3m+3)|H|

– start using Z as shorthand for \((i_1, i_2, i_3, b_1, b_2, b_3)\)

– another way to write \(a'\)'s a':

\[ p_{a}(Z) = 0 \quad \forall Z \in (F_q)^{3m+3} \]

\[ p_{a}(Z) = 0 \quad \forall Z \in H^{3m+3} \]

\(\text{Proof: same.}\)

\(\text{given:}\)

Focus on another way to write \((F_q)^{3m+3} \rightarrow F_q\) of degree \(\leq 2(3m+3)|H|\)

\(\bullet \) exists \(p_0(F_q)^{3m+3} \rightarrow F_q\) with degree \(\leq 2(3m+3)|H|\)

\(\bullet \) \(p_0(Z) = p_a(Z)\) \(\forall Z \in (F_q)^{3m+3}\)

\(\bullet \) \(p_0(Z) = 0\) \(\forall Z \in H^{3m+3}\)

NP \subseteq \text{PCP}[\log n, \text{polylog } n]

– Focus on \(p_0(Z) = 0 \forall Z \in H^{3m+3}\)

– given: \(p_0(F_q)^{3m+3} \rightarrow F_q\)

– define: \(p_1(x_1, x_2, x_3, \ldots, x_{3m+3}) = \Sigma_{h \in H} p_1(h, x_2, x_3, \ldots, x_{3m+3}) x_i^j\)

– Claim:

\(p_0(Z) = 0 \forall Z \in H^{3m+3} \iff p_1(Z) = 0 \forall Z \in (F_q)^2 \times H^{3m+3}\)

– Proof (\(\Rightarrow\)) for each \(x_2, x_3, \ldots, x_{3m+3} \in H^{3m+3}\), resulting univariate poly in \(x_1\) has all 0 coeffs.

NP \subseteq \text{PCP}[\log n, \text{polylog } n]

– given: \(p_1(F_q)^{3m+3} \rightarrow F_q\)

– define: \(p_2(x_1, x_2, x_3, \ldots, x_{3m+3}) = \Sigma_{h \in H} p_2(h, x_2, x_3, \ldots, x_{3m+3}) x_1^j\)

– Claim:

\(p_1(Z) = 0 \forall Z \in (F_q)^2 \times H^{3m+3}\)

\(\iff\)

\(p_2(Z) = 0 \forall Z \in F_q \times H^{3m+3}\)

– Proof: same.

NP \subseteq \text{PCP}[\log n, \text{polylog } n]

– given: \(p_i : (F_q)^{3m+3} \rightarrow F_q\)

– define: \(p_i(x_1, x_2, x_3, \ldots, x_{3m+3}) = \Sigma_{h \in H} p_i(h, x_2, x_3, \ldots, x_{3m+3}) x_i^j\)

– Claim:

\(p_i(Z) = 0 \forall Z \in (F_q)^{3m+3} \iff p_i(Z) = 0 \forall Z \in F_q \times H^{3m+3}\)

– Proof: same.

NP \subseteq \text{PCP}[\log n, \text{polylog } n]

– define degree \(3m+3+2\) poly, \(\delta_i: F_q \rightarrow F_q\) so that

\(\bullet \) \(\delta_i(v) = 1\) if \(v = i\)

\(\bullet \) \(\delta_j(v) = 0\) if \(0 \leq v \leq 3m+3+1\) and \(v \neq i\)

– define \(Q: F_q \times (F_q)^{3m+3} \rightarrow F_q\) by:

\(Q(v, Z) = \Sigma_{i=0, \ldots, 3m+3} \delta_i(v) p_i(Z) + \delta_{3m+3+1}(v) a^i(Z)\)

– note: degree of \(Q\) is at most \(3(3m+3)|H| + 3m + 3 + 2 < 10m|H|\)
**NP \subseteq PCP[\log n, \text{polylog } n]**

- Recall: MAX-k-PCS gap problem:
  - given:
    - variables $x_1, x_2, \ldots, x_n$ taking values from field $F_q$
    - $n = q^m$ for some integer $m$
    - $k$-ary constraints $C_1, C_2, \ldots, C_r$
  - assignment viewed as $f(F_q)^n \rightarrow F_q$
  - YES: some degree $d$ assignment satisfies all constraints
  - NO: no degree $d$ assignment satisfies more than $(1-\epsilon)$ fraction of constraints

- Instance of MAX-k-PCS gap problem:
  - set $d = 10m|H|$
  - given assignment $Q: F_q \times (F_q)^{3m+3} \rightarrow F_q$
  - expect it to be formed in the way we have described from an assignment $a: H^n \rightarrow \{0,1\}$ to $\varphi$
  - note:
    - to access $a'(Z)$, evaluate $Q(3m+3+1, Z)$
    - $p_a(Z)$ formed from $a'$ and $\psi$ (formed from $\varphi$)
    - to access $p_a(Z)$, evaluate $Q(I, Z)$

- Proof of Lemma (summary):
  - reducing 3-SAT to MAX-k-PCS gap problem
  - $\varphi(x_1, x_2, \ldots, x_n)$ instance of 3-SAT
  - set $m = \Omega(\log n/\log \log n)$
  - $H \subseteq F_q$ such that $|H|^m = n$ ($|H| = \text{polylog } n, q = |H|^3$)
  - generate $(F_q)^{3m+3} = \text{poly}(n)$ constraints:
    - $C_2 = \bigwedge_{(x_1, \ldots, x_n) \in H^n} C_2$
    - each refers to assignment poly $Q$ and $\varphi$ (via $p_a$)
    - all polys $d = O(m|H|) = \text{polylog } n$
    - either all are satisfied or at most $d|q = o(1) \ll \epsilon$

- Key: all low-degree polys

- Schwartz-Zippel: if any one of these sets of constraints is violated at all then at least a $(1 - 12m|H|/q)$ fraction in the set are violated

- $O(\log n)$ random bits to pick a constraint
- query assignment in $O(\text{polylog}(n))$ locations to determine if constraint is satisfied
  - completeness $1$
  - soundness $(1-\epsilon)$ if prover keeps promise to supply degree $d$ polynomial
  - prover can cheat by not supplying proof in expected form
**Definition:** functions $f, g$ are $\delta$-close if
\[ \Pr_{x}[f(x) \neq g(x)] \leq \delta \]

**Lemma:** $\exists \delta > 0$ and a randomized procedure that is given $d$, oracle access to $f: (\mathbb{F}_q)^m \rightarrow \mathbb{F}_q$
\[ \text{-- runs in } \text{poly}(m, d) \text{ time} \]
\[ \text{-- always accepts if } \deg(f) \leq d \]
\[ \text{-- rejects with high probability if } \deg(f) > d \]

- too much to ask. Why?

**Lemma:** $\exists a$ randomized procedure that is given $x$, oracle access to $f: (\mathbb{F}_q)^m \rightarrow (\mathbb{F}_q)$ that is $\delta$-close to a (unique) degree $d$ polynomial $g$
\[ \text{-- runs in } \text{poly}(m, d) \text{ time} \]
\[ \text{-- uses } O(m \log |\mathbb{F}_q|) \text{ random bits} \]
\[ \text{-- outputs } g(x) \text{ with high probability} \]

**Lemma:** $\exists a$ randomized procedure that is given $x$, oracle access to $f: (\mathbb{F}_q)^m \rightarrow (\mathbb{F}_q)$ that is $\delta$-close to a (unique) degree $d$ polynomial $g$
\[ \text{-- runs in } \text{poly}(m, d) \text{ time} \]
\[ \text{-- uses } O(m \log |\mathbb{F}_q|) \text{ random bits} \]
\[ \text{-- outputs } g(x) \text{ with high probability} \]
NP ⊆ PCP[log n, polylog n]

- idea of proof:
  - restrict to random line \( L \) passing through \( x \)
  - query points along line
  - apply error correction

Putting it all together:
- given \( L \in NP \) and an instance \( x \), verifier computes reduction to MAX-\( k \)-PCS gap problem
- prover supplies proof in form
  \[ f : (F_q)^m \rightarrow (F_q)^n \]
  (plus some other info used for low-degree testing)
- verifier runs low-degree test
  - rejects if \( f \) not close to some low degree function \( g \)
  - verifier picks random constraint \( C_i \); checks if sat. by \( g \)
  - uses self-correction to get values of \( g \) from \( f \)
  - accept if \( C_i \) satisfied; otherwise reject