A New Model: Probabilistically Checkable Proofs

- \( \text{NP} \) represents proofs having a deterministic polynomial-time verifier

- \( \text{IP} \) represents proofs where the polynomial-time verifier has two new resources:
  1. randomness, and 2. interaction

Today we study a new model:

- \( \text{PCP} \) represents proofs where the polynomial-time verifier has two new resources:
  1. randomness, and 2. oracle access to proof
Definition of PCP

Let $P$ be an all-powerful prover and $V$ a ppt oracle algorithm. We say that $(P,V)$ is a PCP system for a language $L$ with completeness error $\varepsilon_c$ and soundness error $\varepsilon_s$ if the following holds:

1. **Completeness**: $\forall x \in L$, for $\pi = P(x)$, $\Pr_p[V^\pi(x;\rho) = 1] \geq 1 - \varepsilon_c$

2. **Soundness**: $\forall x \notin L \forall \pi \Pr_p[V^\pi(x;\rho) = 1] \leq \varepsilon_s$

We call $\Pi$ a “PCP”, and can view it as a “robust encoding” of a witness, which admits verification without reading all its symbols.

For IPs, we care about: round complexity, communication complexity, ...

For PCPs, we have a somewhat different set of parameters:

- $\Sigma$: proof alphabet
- $\ell$: proof length
- $q$: verifier query complexity
- $r$: verifier randomness complexity

[Typically queries to $\Pi$ will be non-adaptive]
Some Special Cases

We wish to understand $\text{PCP}[\varepsilon_0, \varepsilon_5, \Sigma, L, g, r, \ldots]$ in different regimes.
Let's start with some special cases to warm up.
Suppose there is no proof ($q = 0$):

- $\text{PCP}[q = 0, r = 0] = P$ ← if there is no proof and no randomness then the verifier is just a polytime algorithm
- $\text{PCP}[q = 0, r = O(\log n)] = P$ ← logarithmically-many random bits don't help
- $\text{PCP}[q = 0, r = \text{poly}(n)] = \text{BPP}$ ← if there is randomness but no proof then the verifier is just a ppt algorithm

Suppose there is no randomness ($r = 0$):

- $\text{PCP}[q = \text{poly}(n), r = 0] = \text{NP}$ ← verifier can read in full a poly-size witness

We denote by $\text{PCP}$ the complexity class with no restrictions beyond "$V$ is ppt". This means that $q = \text{poly}(n)$, $r = \text{poly}(n)$ and allows for $L = \exp(n)$, $|\Sigma| = \exp(n)$. 
Questions

- Which languages have PCPs (beyond NP & BPP)?
  - more than PSPACE

- Do PCPs have benefits for NP languages?
  - (E.g. query complexity sublinear in witness size)
  - yes

- Do PCPs have benefits for tractable languages?
  - (E.g. PCP verification faster than execution)
  - yes

- Are there 2K PCPs for NP languages?
  - yes

Many good news!

But the PCP model is weird (PCP verifier has oracle access to a large proof). How are PCPs useful?

1. lead to interactive arguments (and other crypto proofs) with strong efficiency features
2. lead to hardness & approximation results
Delegation of Computation via PCPs

In the next few lectures, we will work our way up to this result:

**Theorem:** Every language \( L \in \text{NTIME}(T) \) has a PCP where:
- proof length \( \ell = \text{poly}(T) \)
- query complexity \( q = \text{polylog}(T) \)
- prover time \( pt = \text{poly}(T) \)
- verifier time \( vt = \text{poly}(n, \log T) \)

![Diagram](image)

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

**But how to use this "setup"?**

Checking Computations in Polylogarithmic Time

László Babai\(^1\)
Univ. of Chicago\(^6\) and
Eötvös Univ., Budapest

Lance Fortnow\(^2\)
Dept. Comp. Sci.
Univ. of Chicago\(^6\)

Leonid A. Levin\(^3\)
Dept. Comp. Sci.
Boston University\(^4\)

Mario Szegedy\(^5\)
Dept. Comp. Sci.
Univ. of Chicago\(^6\)
A Crypto Interlude: From PCP to Interactive Arguments

**Theorem (informal)**

Suppose $L$ has a PCP with prover time $pt$, verifier time $vt$, query complexity $q$. Then by using cryptography we can construct an interactive "proof" for $L$ s.t.

- prover time $O(pt)$
- verifier time $O(vt)$
- communication $O(q)$.

If we apply this to PCPs in prior slide, we get a powerful result:

\[
\begin{array}{c}
\text{x} \in \Sigma^m \rightarrow \text{poly}(T)-\text{time} \\
\text{prover} \\
\text{we} \in \Sigma^T \rightarrow \text{poly}(n,\log T)-\text{time} \\
\text{verifier} \\
\rightarrow 0/1
\end{array}
\]

**Proof attempt:**

[does not contradict limitations of IPs with small communication!]

\[ P(x,ω) \]

- produce PCP string: $Π := P_{\text{PCP}}(x,ω)$
- deduce query set $Q$ in $V_{\text{PCP}}(x;ρ)$

\[ V(x) \]

- sample PCP randomness $ρ$
- $V_{\text{PCP}}(x;ρ) \equiv 1$

**Problem:** prover can pick $T$ based on $Q$.

[Also, where is the crypto??]
A Crypto Interlude: Kilian’s Protocol

Idea: commit to PCP string first then locally open locations & it

Def: A function family \( H_\lambda = \{ h_\lambda : \{0,1\}^* \rightarrow \{0,1\}^* \} \) is collision-resistant if

\[ \forall \text{efficient adversary } A \quad P_\lambda \left[ \tilde{\Delta}(h) \text{ outputs } x \neq y \text{ s.t. } h(x) = h(y) \right] \text{ is negligible in } \lambda. \]

The new protocol is as follows:

\( P(x,w) \)
- produce PCP string: \( \Pi := \Pi_{PCE}(x,w) \)
- commit to it: \( r^* = MT_h(\Pi) \)
- deduce query set \( Q \) in \( V_{PCE}(x,p) \)
- produce auth paths for each answer

\[ \text{time} (P_{PCE}) + O(\lambda) \]

\( V(x) \)
- sample PCE randomness \( \rho \)
- \( \Pi_{\lambda, \rho, \text{auth}} \)
- \( V_{PCE}(x,p) = 1 \) \& check auth

\[ \text{time} (V_{PCE}) + O(\lambda) (9 \log \lambda) \]

Security analysis involves cryptography and so we will not discuss it.
Upper Bound on PCPs

**Theorem:** $\text{PCP} \subseteq \text{NEXP}$

**Lemma:**
1. $\ell \leq 2^q$ for non-adaptive verifiers
2. $\ell \leq 2^q 121^q$ for adaptive verifiers

[In constructions, $\ell$ is usually smaller than these upper bounds]

**Proof of (i):** there are at most $2^q$ different query sets
**Proof of (ii):** each answer from the proof can lead to a different next query

**Lemma:** $\text{PCP} \left[ \ell, r \right] \subseteq \text{NTIME} \left( (2^q + \ell) \cdot \text{poly}(n) \right)$

**Proof:** Suppose $(P, V)$ is a PCP system for $L$ where the PCP verifier uses $r$ random bits to query a proof of length $\ell$. Consider this decider:

$$D(x, \pi) := \text{For every } \rho \in \{0,1\}^r \text{ compute } b_\rho := V^\pi(x, \rho) \text{ and output }$$

$$\frac{1}{\ell} \sum_{\rho} b_\rho 1_{2^\ell} \iff \sum_{\rho} b_\rho 1_{2^\ell} \geq 1 - 3c.$$

If $x \in L$ then $\exists \pi$ s.t. $D(x, \pi) = 1$. If $x \notin L$ then $\forall \pi D(x, \pi) = 0.$

*dotted section*

- $\Sigma$: proof alphabet
- $\ell$: proof length
- $q$: verifier query complexity
- $r$: verifier randomness complexity
A Simple Inclusion: PSPACE

**Theorem:** $\text{PSPACE} \subseteq \text{PCP}$

**Lemma:** $\text{IP} \subseteq \text{PCP}$

**Proof:** Suppose that $(P,V)$ is a public-coin IP for $L$. (Public coin comes with $L$.) Consider proofs in this format: $\Pi = (a_1, z_1, u \notin \exists a_2, z_2 \forall r_2 \ldots u \notin \exists a_n, z_n \forall r_n \ldots r_k)$. The PCP verifier samples $r_1, \ldots, r_k$ and accepts if the IP verifier accepts:

$$V (x, a_1, a_2, \ldots, a_n, r_1, \ldots, r_k) = 1.$$ 

**Completeness:** Consider the honest proof $\Pi := (P(x), r_1, P(x), \ldots, P(x), r_k)$. 

**Soundness:** Any proof in the above format corresponds to an "unrolled" IP proof.

**In sum:** $\text{PSPACE} \subseteq \text{PCP} \subseteq \text{NEXP}$. We will see that $\text{PCP} = \text{NEXP}$ by recycling techniques (arithmetization, sumcheck) and using new ones (low-degree testing). We will also see how to "scale down" to get PCPs for NP.