Today

- IP = PSPACE.
  - IP for straightline programs.
  - Straightline program for PSPACE.
- Other Proof systems and known results.
  - Multiprover IP (MIP): MIP = NEXP
  - Prob. checkable proof: PCP = NP.
- PCP and consequence to inapproximability.

Last lecture

- Showed \#P has IP proofs.
- abstracted to ”sequences of polynomials”.
- today: will recall abstraction in simpler form.

Straightline program of polynomials

- $L, n, d, w$ straightline program of polynomials is a sequence of polynomials $P_0, P_1, \ldots, P_L$
each on $n$ variables of degree at most $d$, with $P_i$ being polytime computable by making $w$ calls to oracle for $P_{i-1}$.
- For simplicity, assume $P_i(x)$ computed by adding/multiplying value of $P_{i-1}(f_i(x))$
$P_{i-1}(g_i(x))$, where $f_i$’s and $g_i$’s are polynomial time computable (so $w = 2$).

Lines in $\mathbb{Z}_p^n$

A line $\ell_{x,y}$ through $x, y \in \mathbb{Z}_p^n$ is the set of points $\{\ell_{x,y}(t) | t \in \mathbb{Z}_p\}$ where $\ell_{x,y}(t) = (1 - t)x + t \cdot y$.

Function $P$ restricted to line $\ell$ is the function $P|_{\ell}(t) = P(\ell(t))$.

If $P$ has degree $d$, then $P|_{\ell}$ has degree $d$. 
Given straightline program \( \{P_0, \{f_i\}, \{g_i\}, \{\sigma_i\}\} \), where \( \sigma_i \in \{+, *\} \), here’s how to prove \( P_L(z) = a \).

- **Iteration \( L - i \):** Claiming \( P_i(z_i) = a_i \).
- Let \( \ell_i = \ell_{f_i(z_i)}g_i(z_i) \).
- \( P \rightarrow V : h_i \), a univariate polynomial of degree \( \leq d \) (supposedly \( h_i = P_{i-1}|_{\ell_i} \)).
- \( V: \) If \( h_i(0)\sigma_i h_i(1) \neq a_i \), REJECT, else pick \( t_i \in \mathbb{Z}_p \) at random and set \( z_{i-1} = \ell_i(t_i) \) and \( a_{i-1} = h_i(t_i) \) and send \( z_{i-1}, a_{i-1} \) to Prover.
- **Final iteration:** Compute \( P_0(z_0) \) on one’s own.

### Analysis

- Completeness: Prover just sends \( h_i = P_i|_{\ell_i} \) in each iteration and will be accepted w.p. 1.
- Soundness: As in previous proof: If \( P_i(z_i) \neq a_i \), and verifier does not REJECT, then w.p. \( 1 - d/p \), \( P_{i-1}(z_{i-1}) \neq a_{i-1} \).
- Conclude: Have IP for straightline program value.

### Straightline program for PSPACE

- **Idea:** Let \( x, y \) be binary strings denoting configuration of PSPACE machine \( M \).
- \( P_i(x, y) = 1 \) iff go from config. \( x \) to \( y \) in \( 2^i \) steps.
- So \( P_n(x_0, x_{\text{acc}}) = 1 \) is PSPACE-complete.
- \( P_i(x, y) = \sum_z P_{i-1}(x, z) \cdot P_{i-1}(z, y) \).  
  Almost works, except sums of exponentially many terms. So break sum down.
  - Let \( Q_{i,n}(x, y, z) = P_{i-1}(x, z)P_{i-1}(z, y) \)
  - \( Q_{i,j}(x, y, z_1, \ldots, z_j) = Q_{i,j+1}(x, y, z_1, \ldots, z_j) \)
  - \( P_i(x, y) = Q_{i,0}(x, y) \).
Other models of proof systems

- Multiprover proofs: What if there are two non-interacting provers that verifier can quiz?
  - Potentially more powerful.
  - Indeed a priori can only show MIP in \( \text{NEXPTIME} \).
- Oracle interactive proof: Oracle fixed - what can you prove.
  - Simulates MIP, but can be simulated by two provers.
- Probabilistically Checkable Proof
  - Usual proof string, with random access.

Verifier randomized, but restricted \# of probes/queries into proof.
- How powerful? PCP = OIP!

---

Some results

- MIP = 2IP = NEXPTIME. Inspired by \( \text{IP} = \text{PSPACE} \). Indeed can describe NEXP as \( \exists P_0 \) s.t. \( P_L(z) = a \).
- Let PCP\([r,q]\) be things you can proof with prob. polytime verifier tossing \( r(n) \) coins and querying proof \( q(n) \) times with completeness 1 and soundness \( 1/2 \).
- MIP = NEXPTIME implies \( \text{NP} \subseteq \text{PCP[polylog, polylog]} \).
- But with lots of more work \( \text{NP} = \text{PCP[O(log), 3]} \).

---

Consequence to inapproximability