Interactive Oracle Proofs

Recall that NP is the model for traditional mathematical proofs:

![Diagram of Prover and Verifier](image)

We have studied two different extensions:

- **IP**: add randomness & interaction
- **PCP**: add randomness & oracle access to proof

Today we consider the common extension between the two:

**Interactive Oracle Proof (IOP)**
add randomness, interaction, and oracle access to proof

![Diagram of Interactive Oracle Proof](image)
Definition of IOP

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

1. **Completeness**: \( \forall x \in L \quad \Pr[P(x), V(x; r) = 1] \geq 1 - \epsilon_c \)

2. **Soundness**: \( \forall x \notin L \quad \forall v \quad \Pr[P, V(x; r) = 1] \leq \epsilon_s \)

Above \( <A, B> \) denotes the process: \( A \rightarrow \pi_1, m_1 \in B^{\pi_1}, A(m_1) \rightarrow \pi_2, m_2 \in B^{\pi_1, \pi_2}, \) and so on until \( B \) decides to halt and output.

**Efficiency measures:**

- prover time
- verifier time
- round complexity
- randomness complexity
- alphabet size
- proof length \( (l_{\pi_1} + l_{\pi_2} + \ldots) \)
- query complexity \( (q_1 + q_2 + \ldots) \)

- public vs. private coins

Each verifier message is random, so all queries can be at the end [interaction phase, then query phase]
Upper Bound and Lower Bound

Let IOP be the set of languages decidable via an interactive oracle proof.

**Lemma:** $\text{NP} \subseteq \text{IOP}$

**proof:** We have proved that $\text{NP} \subseteq \text{PCP}$ and a PCP is a special case of an IOP:

$$\text{PCP}[\mathcal{E}_c, \mathcal{E}_s, \Sigma, l, q, r, \ldots] \subseteq \text{IOP}[\mathcal{E}_c, \mathcal{E}_s, k=0.5, \Sigma, l, q, r, \ldots].$$

You can think that "NP is to IP like PCP is to IOP".

**Lemma:** $\text{IOP} \subseteq \text{NEXP}$

**proof:** We have proved that $\text{PCP} \subseteq \text{NEXP}$, and any IOP can be "unrolled" into a (very long) PCP, analogously to how we unrolled an IP into a PCP.

That is:

$$\text{IOP}[\mathcal{E}_c, \mathcal{E}_s, K, \Sigma, (l_p, l_w), \ldots] \subseteq \text{PCP}[\mathcal{E}_c, \mathcal{E}_s, \Sigma, k=(|\Sigma|^{l_p})^l_p].$$

The maximum PCP proof length is $2^{\text{poly}(n)} \cdot \text{exp}(n) = \text{exp}(n)$.

We conclude that $\text{IOP}=\text{NEXP}$. 
What are IOPs good for?

We have learned that IOPs do not give us new languages over PCPs. This is ok: we can try to achieve better parameters for languages in NEXP.

Our goal: leverage interaction to design IOPs that are "more efficient" (shorter proof length, fewer queries, etc.) than state-of-the-art PCPs.

But... PCPs were an awkward proof model and IOPs are only more awkward. So why care about the goal?

Similarly to PCPs, we can use cryptography to compile IOPs into cryptographic proofs (aka arguments). And if we can design efficient IOPs then we will get cryptographic proofs that are more efficient than from PCPs!

In the next few lectures we will learn how to construct IOPs that achieve parameter regimes that we do not know how to achieve with PCPs.

Curiously, despite this, to date we do not have strong separations between IOPs & PCPs.
From IOP to Interactive Argument

**Theorem [informal]**

Suppose $L$ has a public-coin IOP with prover time $pt$, verifier time $vt$, query complexity $q$. Then by using cryptography we can construct an interactive argument for $L$ with

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

**Proof attempt:**

1. deduce all the oracles:
   \[ \Pi_i = P_{\text{top}}(x), \Pi_2 = P_{\text{top}}(x, r_1), \ldots, \Pi_k = P_{\text{top}}(x, r_1, \ldots, r_k) \]

2. deduce IOP verifier's queries:
   \[ Q = \text{queries (V_{\text{top}}(x, r_1, \ldots, r_k))} \]

\[ V(x) \]
\[ \Pi_i, \ldots, \Pi_k \]

This is NOT secure because the prover can answer queries based on $r_1, \ldots, r_k$!

**Idea:** extend Kilian's protocol from PKE to IOP by committing to each oracle via a Merkle tree and then locally open the relevant locations.
From IOP to Interactive Argument

As in Kilian’s protocol, we rely on collision-resistant functions to build Merkle trees.

\[ P(x) \]

\[ \pi_1 := \pi_{IOP}(x), \; r_{I1} := MT_h(\pi_1) \]

\[ \pi_2 := \pi_{IOP}(x, r_1), \; r_{I2} := MT_h(\pi_2) \]

\[ \pi_k := \pi_{IOP}(x, r_1, \ldots, r_{k-1}), \; r_{Ik} := MT_h(\pi_k) \]

- deduce IOP verifier’s queries:
  \[ \pi := \text{queries}(V_{IOP}^{\pi_{IOP}}; \pi_k(x; r_1, \ldots, r_k)) \]
- produce output paths for each answer

\[ \text{time}(P) = \text{time}(P_{IOP}) + O_x(l) \]

\[ V(x) \]

\[ h \rightarrow r_{I1} \rightarrow r_1 \rightarrow r_{I2} \rightarrow r_2 \quad \vdots \quad \rightarrow r_k \rightarrow r_k \rightarrow \text{ans, paths} \rightarrow \]

\[ V_{IOP}(x; r_1, \ldots, r_k) \equiv 1 \& \text{check paths} \]

\[ O_x(9\log l) \quad \text{time}(V) = \text{time}(V_{IOP}) + O_x(9\log l) \]

Security analysis involves cryptography and so we will not discuss it.

In sum, designing efficient IOPs leads to efficient arguments.
Recycling: IOP for NTIME from the PCP for NTIME

$P((m,n,\phi,\varepsilon),A)$

1. Compute $C := T(\mathcal{H}, H, (m,n))$
2. Output $\Pi_A : \mathbb{F}^n \rightarrow \mathbb{F}$ that equals the $(\mathcal{H}, H, \bar{n})$-extension of $A$: $\{0,1\}^n \rightarrow \{0,1\}$
3. Do these sumchecks in parallel:

$P_{\text{sc}}(\mathcal{H}, H, \bar{n}, o)$

$V_{\text{sc}}(\mathcal{H}, H, \bar{n}, o)$

$V((m,n,\phi,\varepsilon))$

1. Compute $C := T(\mathcal{H}, H, (m,n))$
2. Low-degree test $\Pi_A$ for individual degree < $1H_1$
   or total degree < $n \cdot 1H_1$
3. Do these sumchecks in parallel:

$P_{\text{sc}}(\mathcal{H}, H, \bar{n}, 0)$

$V_{\text{sc}}(\mathcal{H}, H, \bar{n}, 0)$

Boolean

Constraints

Inputs 8
Analysis

If $F$ has size at least $|H| \cdot \text{poly}(|\phi|)$ then the protocol is sound:

$$\varepsilon_s \leq \frac{\varepsilon_{\text{LDT}}(s)}{H_1} + O(\delta) + O\left(\frac{n \cdot |H|}{H_1}\right) + O\left(\frac{(n+3m) \cdot |H_1| \cdot |\phi|}{H_1}\right) + O\left(\frac{n \cdot |H|}{H_1}\right) \leq O(1)$$

Moreover, if $|F| = |H| \cdot \text{poly}(|\phi|)$ and $|H| = |\phi|^{1/3}$ then the protocol is efficient:

- **Proof length:**
  \[
  |\Pi_A| + |\Sigma_1| + |\Sigma_2| + |\Sigma_3| = |F| \cdot n + O(n \cdot |H|) + O((m+3n) \cdot |H_1| \cdot |\phi|)
  = |F| \cdot \frac{\log |H_1| + O(\log |\phi|)}{\log |H_1|} \cdot n = 2 \cdot \frac{\log |H_1| + O(\log |\phi|)}{\log |H_1|} \cdot n = (2^n) \cdot O(1)
  \]

- **Query complexity:**
  \[q_{\text{LDT}} + O(1) + O(n \cdot |H|) + O((m+3n) \cdot |H_1| \cdot |\phi|) + O(n \cdot |H|) = O((m+n) \cdot |H_1| \cdot |\phi|) = |\phi|^{0.17}
  \]

- **Verifier time:**
  \[t_{\text{LDT}} + \text{poly}(n,|H|) + \text{poly}(m+3n,|H_1|,|\phi|) + \text{poly}(n,|H_1|,|\phi|) = \text{poly}(|\phi|^{1/3},|\phi|)
  \]

The reduction from $\text{NTIME}(T)$ to $\text{OSAT}$ can be improved to achieve

\[n = \log T + O(\log \log T), \quad m = O(\log T), \quad |\phi| = \text{poly}(\log T)
\]

which yields

\[l = T^{1+O(\varepsilon)}, \quad q = (\log T)^{O(1/\varepsilon)}, \quad p_l = \text{poly}(T), \quad v_T = \text{poly}(|x|, (\log T)^{1/3})
\]
Towards Efficient IOPs

We have shown (up to the improved reduction from NTIME(T) to OsAT) that

\[ \text{NTIME}(T) \leq \text{IoP} \left[ \varepsilon = 0, \varepsilon_0 = 0.5, T(n) = \Omega(n) \text{ and } \forall \varepsilon > 0 \right] \]

\[ \begin{align*}
  l &= T^{1+O(1)}, \\
  q &= (\log T)^{O(1)}, \\
  r &= \text{poly}_n(\log T)
\end{align*} \]

Without much effort, we reduced proof length significantly!

Q: can we reduce proof length even further (e.g. to linear)?

A serious obstacle to improving proof length is that we are encoding assignments via the multi-variate low-degree extension (also known as the Reed-Muller code), which inherently incurs a polynomial blowup:

\[ |\text{FI}| \geq (n \cdot \log H) \left( \frac{2^{\log H_1 + \log n - \log \log H_1}}{\log H_1} \right) \cdot n = (2^n) \left( 1 + \frac{\log n - \log \log H_1}{\log H_1} \right) = (2^n)^{1+O(1)} \]

To do better, we will change how we encode assignments.

Reason for optimism: we are severely underusing the IOP model, as the prover sends a proof oracle in the first round only. We should send oracles in more rounds!