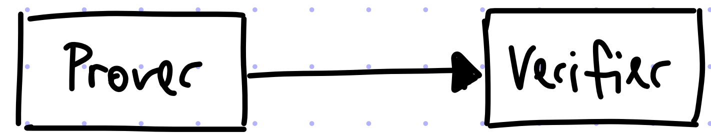
Lecture 15

Foundations of Probabilistic Proofs
Fall 2020
Alessandro Chiesa

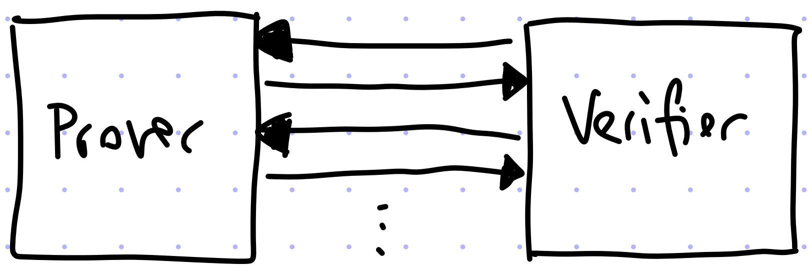
Interactive Oracle Proofs

Recall that NP is the model for traditional mathematical proofs:

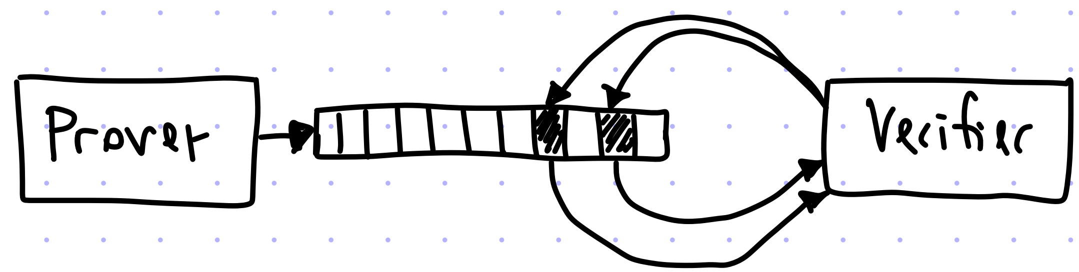


We have studied two different extensions:

IP: add randomness
& interaction



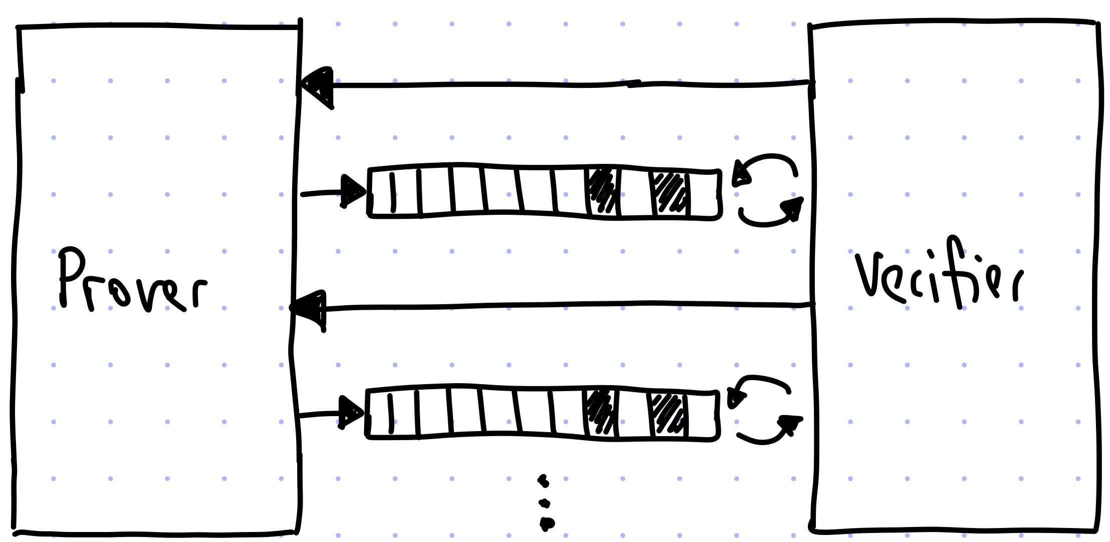
PCP: add randomness
& oracle access to proof



Today we consider the common extension between the two:

Interactive Oracle Proof (IOP)

add randomness, interaction, and oracle access to proof



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 ε_c and soundness error ε_s if the following holds:

① completeness: $\forall x \in L \ \Pr_{\rho}\left[\langle P(x), V(x;\rho)\rangle = 1\right] \geq 1 - \varepsilon_c$

② soundness: $\forall x \notin L \ \forall \tilde{P} \ \Pr_{\rho}\left[\langle \tilde{P}, V(x;\rho)\rangle = 1\right] \leq \varepsilon_s$

Above $\langle A,B\rangle$ denotes this process: $A \to \pi_1$, $m_1 \leftarrow B^{\pi_1}$, $A(m_1) \to \pi_2$, $m_2 \leftarrow B^{\pi_1,\pi_2}$, and so on until B decides to halt and output.

Efficiency measures:

- prover time
- verifier time
- round complexity
- randomness complexity

- alphabet size
- proof length $(|\pi_1| + |\pi_2| + \dots)$
- query complexity $(q_1 + q_2 + \dots)$

- 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: NEXP $\subseteq$ IOP

proof: We have proved that NEXP $\subseteq$ PCP and a PCP is a special case of an IOP:

$$PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r, \ldots] \subseteq IOP[\varepsilon_c, \varepsilon_s, k=0.5, \Sigma, \ell, q, r, \ldots].$$

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

lemma: IOP $\subseteq$ NEXP

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

That is: $IOP[\varepsilon_c, \varepsilon_s, K, \Sigma, (\ell_p, \ell_v), \ldots] \subseteq PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell = (|\Sigma|^{\ell_v}) \cdot \ell_p]$.

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

We conclude that IOP = 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 [1/2]

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:

P(x)

1. deduce all the oracles:

   $\pi_1 := P_{IOP}(x)$, $\pi_2 := P_{IOP}(x, r_1)$, …, $\pi_k := P_{IOP}(x, r_1, \dots, r_{k-1})$

2. deduce IOP verifier's queries:

   $Q := \text{queries}\left(V_{IOP}^{\pi_1,\dots,\pi_k}(x; r_1,\dots,r_k)\right)$

$\xleftarrow{\; r_1, \dots, r_k \;}$

$\xrightarrow{\; [\pi_1,\dots,\pi_k]|_Q \;}$

V(x)

$V_{IOP}^{[\pi_1,\dots,\pi_k]|_Q}(x; r_1,\dots,r_k) \stackrel{?}{=} 1$

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

Idea: extend Kilian's protocol from PCP 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.

$P(x)$		$V(x)$
	$\xleftarrow{\;h\;}$	sample CRH: $h \leftarrow H_\lambda$
$\pi_1 := P_{IOP}(x)$, $rt_1 := MT_h(\pi_1)$	$\xrightarrow{\;rt_1\;}$	
	$\xleftarrow{\;r_1\;}$	
$\pi_2 := P_{IOP}(x, r_1)$, $rt_2 := MT_h(\pi_2)$	$\xrightarrow{\;rt_2\;}$	
	$\xleftarrow{\;r_2\;}$	
	$\vdots$	
$\pi_k := P_{IOP}(x, r_1, \dots, r_{k-1})$, $rt_k := MT_h(\pi_k)$	$\xrightarrow{\;rt_k\;}$	
	$\xleftarrow{\;r_k\;}$	
• deduce IOP verifier's queries: $Q := \text{queries}\left(V_{IOP}^{\pi_1,\dots,\pi_k}(x; r_1, \dots, r_k)\right)$ • produce auth paths for each answer	$\xrightarrow{\;\text{ans, paths}\;}$	$V_{IOP}^{\text{ans}}(x; r_1, \dots, r_k) \stackrel{?}{=} 1$ & check paths

$\text{time}(P) = \text{time}(P_{IOP}) + O_\lambda(\ell)$

$\underbrace{\qquad\qquad}_{O_\lambda(q \log \ell)}$

$\text{time}(V) = \text{time}(V_{IOP}) + O_\lambda(q \log \ell)$

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

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

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

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

1. Compute $C := T(\mathbb{F},H,(m,n,\phi))$
2. low-degree test Π_A for individual degree $< |H|$ (or total degree $< \bar{n}\cdot|H|$)
3. Do these sumchecks in parallel:

Π_A ⋮ LDT

[4 queries at random locations from the sumchecks]

$r_1,\dots,r_{\bar{m}+3\bar{n}} \in \mathbb{F}$

booleanity

$P_{sc}(\mathbb{F},H,\bar{n},0)$

$$\sum_{a\in H^{\bar{n}}} \Pi_A(a)(1-\Pi_A(a))\prod_{i\in[\bar{n}]}\hat{r}_i(a_i) \stackrel{?}{=} 0$$

$V_{sc}(\mathbb{F},H,\bar{n},0)$

$(s_1,\dots,s_{\bar{n}})$
- query Π_A at $(s_1,\dots,s_{\bar{n}})$
- for $i=1,\dots,\bar{n}$: eval $\hat{r}_i(X)$ at s_i

constraints

$P_{sc}(\mathbb{F},H,\bar{m}+3\bar{n},0)$

$$\sum_{a=(w,v_1,v_2,v_3)\in H^{\bar{m}+3\bar{n}}} C(w,v_1,v_2,v_3,\Pi_A(v_1),\Pi_A(v_2),\Pi_A(v_3))\prod_{i\in[\bar{m}+3\bar{n}]}\hat{r}_i(a_i)=0$$

$V_{sc}(\mathbb{F},H,\bar{m}+3\bar{n},0)$

$(s_1,\dots,s_{\bar{m}+3\bar{n}})$
- query Π_A at $(s_{\bar{m}+1},\dots,s_{\bar{m}+\bar{n}})$, $(s_{\bar{m}+\bar{n}+1},\dots,s_{\bar{m}+2\bar{n}})$, $(s_{\bar{m}+2\bar{n}+1},\dots,s_{\bar{m}+3\bar{n}})$
- eval C at (s,ans_1,ans_2,ans_3)
- for $i=1,\dots,\bar{m}+3\bar{n}$: eval $\hat{r}_i(X)$ at s_i

input z

$P_{sc}(\mathbb{F},H,\bar{n},0)$

$$\sum_{a\in H^{\frac{\log|z|}{\log|H|}}\times 0^{\bar{n}-\frac{\log|z|}{\log|H|}}} (\Pi_A(a)-\hat{z}(a))\prod_{i\in[\bar{n}]}\hat{r}_i(a_i) \stackrel{?}{=} 0$$

$V_{sc}(\mathbb{F},H,\bar{n},0)$

$(s_1,\dots,s_{\bar{n}})$
- query Π_A at $(s_1,\dots,s_{\bar{n}})$
- eval $\hat{z}$ at $(s_1,\dots,s_{\bar{n}})$
- for $i=1,\dots,\bar{n}$: eval $\hat{r}_i(X)$ at s_i

Analysis

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

$$\varepsilon_s \leq \underbrace{\varepsilon_{LDT}(\delta)}_{\substack{\text{soundness error of LDT} \\ \text{if } \Pi_A \text{ is } \delta\text{-far}}} + \underbrace{O(\delta)}_{\substack{\text{pr that any of } O(1) \text{ queries to } \Pi_A \\ \text{sees an error, if } \Pi_A \text{ is } \delta\text{-close}}} + \underbrace{O\left(\frac{\bar{n}\cdot|H|}{|\mathbb{F}|}\right)}_{\substack{\text{booleanity} \\ \text{sumcheck}}} + \underbrace{O\left(\frac{(\bar{n}+3\bar{m})\cdot(|H|\cdot|\phi|)}{|\mathbb{F}|}\right)}_{\substack{\text{constraints} \\ \text{sumcheck}}} + \underbrace{O\left(\frac{\bar{n}\cdot|H|}{|\mathbb{F}|}\right)}_{\substack{\text{input} \\ \text{sumcheck}}} \leq O(1)$$

Moreover, if $|\mathbb{F}| = |H|\cdot\mathrm{poly}(|\phi|)$ and $|H| = |\phi|^{\frac{1}{\varepsilon}}$ then the protocol is efficient:

- proof length: $|\Pi_A| + |SC1| + |SC2| + |SC3| = |\mathbb{F}|^{\bar{n}} + O(\bar{n}\cdot|H|) + O((\bar{m}+3\bar{n})\cdot|H|\cdot|\phi|)$

$$= |\mathbb{F}|^{\frac{n}{\log|H|}} + O\left(\frac{m+n}{\log|H|}\cdot|H|\cdot|\phi|\right) = (|H|\cdot\mathrm{poly}(|\phi|))^{\frac{n}{\log|H|}} = 2^{\frac{\log|H| + O(\log|\phi|)}{\log|H|}\cdot n} = 2^{\left(1+O\left(\frac{\log|\phi|}{\log|H|}\right)\right)\cdot n} = (2^n)^{1+O(\varepsilon)}$$

- query complexity: $q_{LDT} + O(1) + O(\bar{n}\cdot|H|) + O((\bar{m}+3\bar{n})\cdot|H|\cdot|\phi|) + O(\bar{n}\cdot|H|) = O((m+n)|H|\cdot|\phi|) = |\phi|^{O(\frac{1}{\varepsilon})}$

- verifier time: $t_{LDT} + \mathrm{poly}(\bar{n},|H|) + \mathrm{poly}((\bar{m}+3\bar{n}),|H|\cdot|\phi|) + \mathrm{poly}(\bar{n},|H|,|z|) = \mathrm{poly}(|\phi|^{\frac{1}{\varepsilon}},|z|)$

The reduction from NTIME(T) to OSAT can be improved to achieve

$$n = \log T + O(\log\log T)\ ,\ m = O(\log T)\ ,\ |\phi| = \mathrm{poly}(\log T)$$

which yields

$$\ell = T^{1+O(\varepsilon)},\quad q = (\log T)^{O(\frac{1}{\varepsilon})},\quad pt = \mathrm{poly}(T),\quad vt = \mathrm{poly}(|x|,(\log T)^{\frac{1}{\varepsilon}})$$

Towards Efficient IOPs

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

theorem: For every time function $T:\mathbb{N}\to\mathbb{N}$ with $T(n)=\Omega(n)$ and $\forall \varepsilon>0$

$$\mathrm{NTIME}(T) \subseteq \mathrm{IOP}\begin{bmatrix} \varepsilon_c=0,\ \varepsilon_s=0.5,\ \Sigma=\{0,1\},\ pt=\mathrm{poly}_\varepsilon(T),\ vt=\mathrm{poly}_{1/\varepsilon}(n,\log T) \\ \ell=T^{1+O(\varepsilon)},\ q=(\log T)^{O(\frac{1}{\varepsilon})},\ r=\mathrm{poly}_{1/\varepsilon}(\log T) \end{bmatrix}$$

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:

$$|\mathbb{F}|^{\bar{n}} \geq (\bar{n}\cdot|H|)^{\frac{n}{\log|H|}} = 2^{\frac{\log|H|+\log n-\log\log|H|}{\log|H|}\cdot n} = (2^n)^{\left(1+\frac{\log n-\log\log|H|}{\log|H|}\right)} = (2^n)^{1+O(\varepsilon)}$$

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!