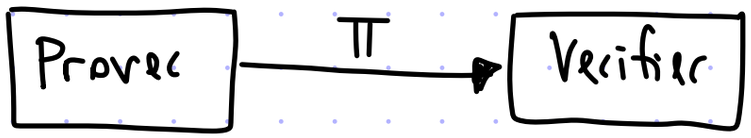
Lecture 08

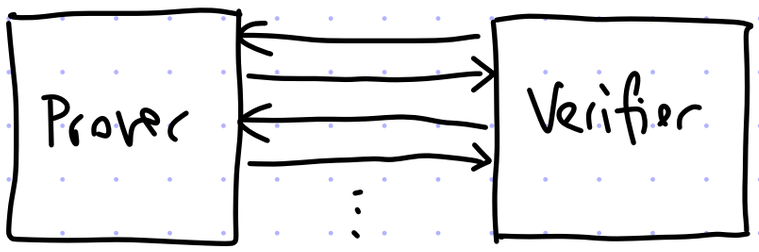
Foundations of Probabilistic Proofs
Fall 2020
Alessandro Chiesa

A New Model: Probabilistically Checkable Proofs

- NP represents proofs having a deterministic polynomial-time verifier

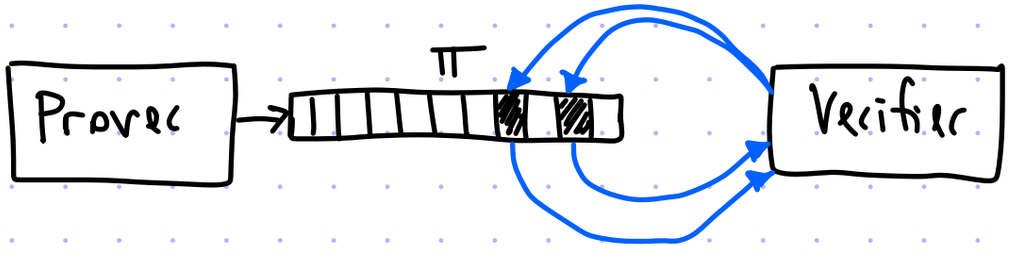


- IP represents proofs where the polynomial-time verifier has two new resources:
  ① randomness, and ② interaction



Today we study a new model:

- 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 ε_c and soundness error ε_s if the following holds:

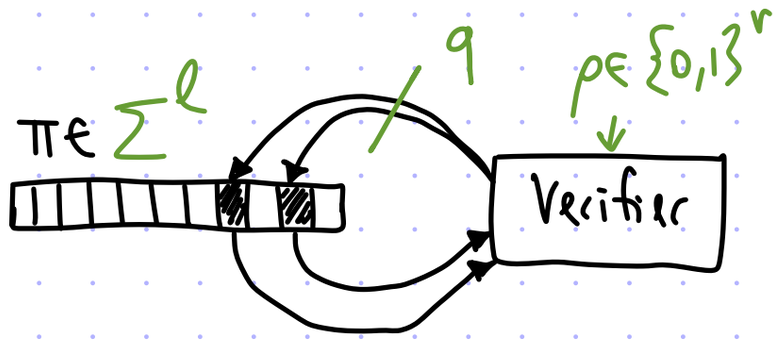
① completeness: $\forall x \in L$, for $\pi := P(x)$, $\Pr_\rho \left[V^\pi(x;\rho) = 1 \right] \geq 1 - \varepsilon_c$

② soundness: $\forall x \notin L \ \forall \tilde{\pi} \quad \Pr_\rho \left[V^{\tilde{\pi}}(x;\rho) = 1 \right] \leq \varepsilon_s$

We call π a "PCP", and can view it as a "robust encoding" of a witness, which admits verification without reading all its symbols.

For IPs we cared about: round complexity, communication complexity, ...
For PCPs we have a somewhat different set of parameters:

- Σ : proof alphabet
- ℓ : proof length
- q : verifier query complexity
- r : verifier randomness complexity



[typically queries to π will be non-adaptive]

Some Special Cases

We wish to understand $PCP[\varepsilon_c, \varepsilon_s, \Sigma, \ell, q, r, \ldots]$ in different regimes.

Let's start with some special cases to warm up.

Suppose there is no proof $(q = 0)$:

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

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

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

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

Questions

- Which languages have PCPs (beyond NP & BPP)? — more than PSPACE
- Do PCPs have benefits for NP languages? — yes
  (E.g. query complexity sublinear in witness size)
- Do PCPs have benefits for tractable languages? — yes
  (E.g. PCP verification faster than execution)
- 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?

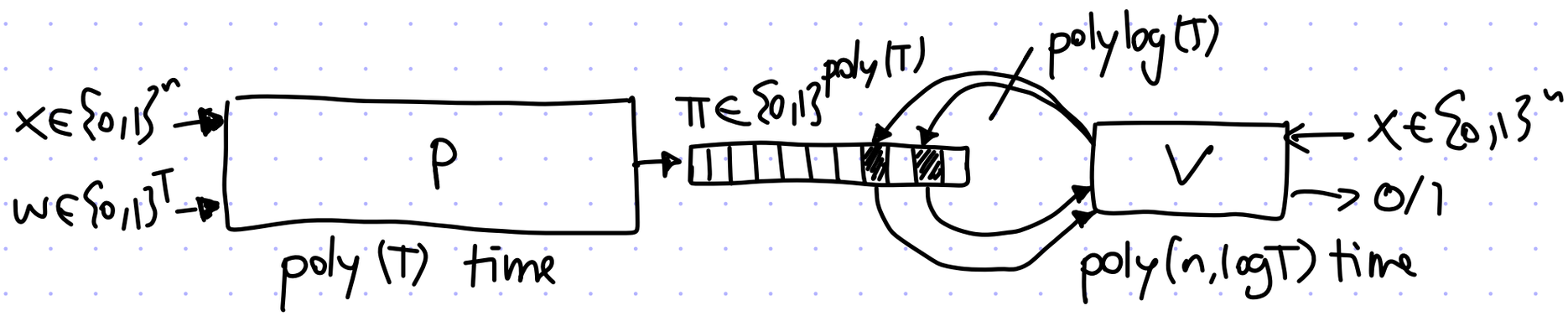
① lead to interactive arguments (& other crypto proofs) with strong efficiency features
② lead to hardness & approximation results

Delegation of Computation via PCPs

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

theorem: Every language $L \in NTIME(T)$ has a PCP where:

proof length	$\ell = poly(T)$	query complexity	$q = polylog(T)$
prover time	$pt = poly(T)$	verifier time	$vt = poly(n, \log T)$



> 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]

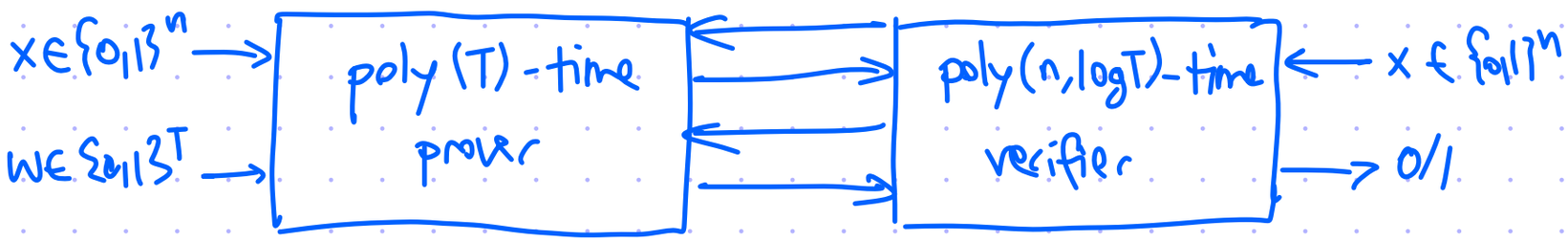
A Crypto Interlude: From PCP to Interactive Arguments

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.

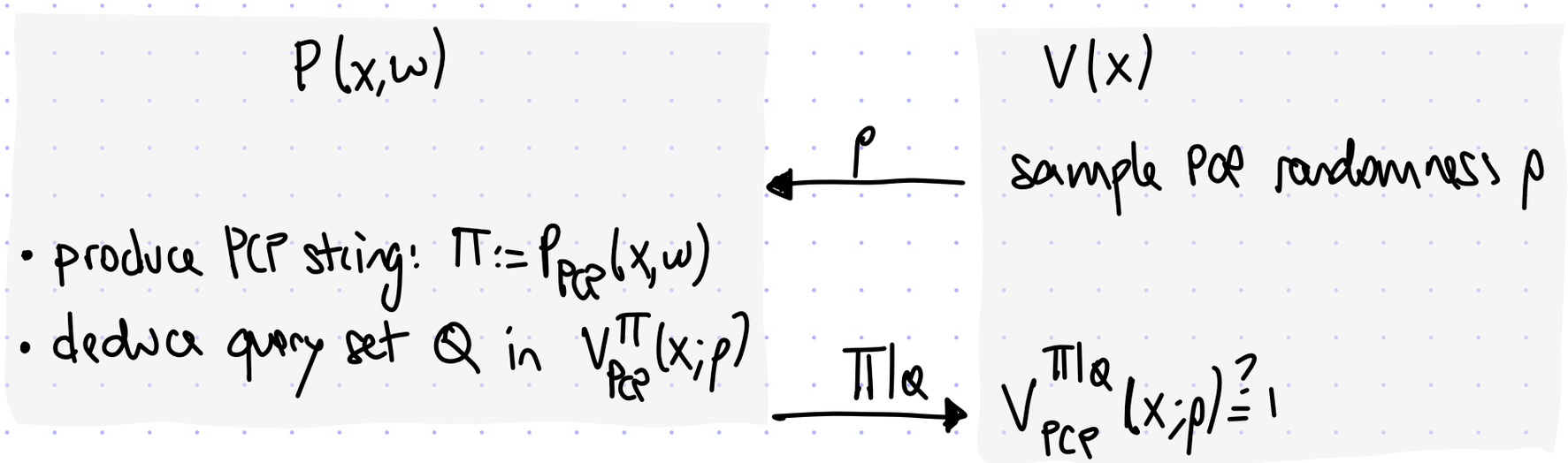
prover time $O(pt)$, verifier time $O(vt)$, communication $O(q)$.

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



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

Proof attempt:



Problem: prover can pick π based on Q.

[Also, where is the crypto??]

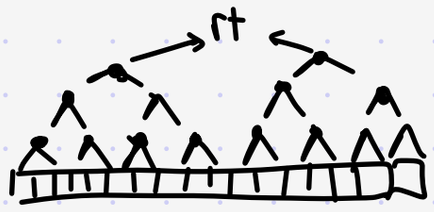
A Crypto Interlude: Kilian's Protocol

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

Def: A function family $H_\lambda = \{h_\lambda : \{0,1\}^{2^\lambda} \to \{0,1\}^\lambda\}$ is **collision-resistant** if

$\forall$ efficient adversary A $\Pr_{h \leftarrow H_\lambda}[A(h) \text{ outputs } x \neq y \text{ s.t. } h(x) = h(y)]$ is negligible in λ.

The new protocol is as follows:

$P(x, w)$		$V(x)$
	$\xleftarrow{h}$	$h \leftarrow H_\lambda$
• produce PCP string: $\pi := P_{PCP}(x, w)$ • commit to it: $rt = MT_h(\pi)$ 	$\xrightarrow{rt}$	
	$\xleftarrow{\rho}$	sample PCP randomness ρ
• decide query set Q in $V_{PCP}^{\pi}(x;\rho)$ • produce auth paths for each answer	$\xrightarrow{\pi\|_Q, \text{auth}}$	$V_{PCP}^{\pi\|_Q}(x;\rho) \stackrel{?}{=} 1$ & check auth
$\text{time}(P_{PCP}) + O_\lambda(\ell)$	$q(q \log \ell)$	$\text{time}(V_{PCP}) + O_\lambda(q \log \ell)$

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

Upper Bound on PCPs

- Σ : proof alphabet
- ℓ : proof length
- q : verifier query complexity
- r : verifier randomness complexity

Theorem: $PCP \subseteq NEXP$

Lemma: (i) $\ell \leq 2^r q$ for non-adaptive verifiers
(ii) $\ell \leq 2^r |\Sigma|^q q$ for adaptive verifiers

[in constructions ℓ is usually smaller than these upper bounds]

proof of (i): there are at most 2^r different query sets

proof of (ii): each answer from the proof can lead to a different next query

lemma: $PCP[\ell, r] \subseteq NTIME((2^r+\ell)\cdot poly(n))$.

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

$D(x, \pi)$:= For every $\rho \in \{0,1\}^r$ compute $b_\rho := V^\pi(x;\rho)$ and output 1 iff $\sum_\rho b_\rho / 2^r \geq 1-\varepsilon_c$.
($x \in \{0,1\}^n$, $\pi \in \{0,1\}^\ell$)

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

A Simple Inclusion: PSPACE

Theorem: $\mathsf{PSPACE} \subseteq \mathsf{PCP}$

Lemma: $\mathsf{IP} \subseteq \mathsf{PCP}$

proof: Suppose that (P,V) is a public-coin IP for L. (Public coin comes w.l.o.g.)
Consider proofs in this format: $\Pi = \{a_{r_1}\}_{r_1} \cup \{a_{r_1,r_2}\}_{r_1,r_2} \cup \cdots \cup \{a_{r_1,\ldots,r_k}\}_{r_1,\ldots,r_k}$.
The PCP verifier samples $r_1,\ldots,r_k$ and accepts if the IP verifier accepts:

$$V\left(x, a_{r_1}, a_{r_1,r_2}, \ldots, a_{r_1,\ldots,r_k}; r_1 \cdots r_k\right) \stackrel{?}{=} 1.$$

Completeness: consider the honest proof $\Pi := \{P(x,r_1)\}_{r_1} \cup \{P(x,r_1,r_2)\}_{r_1,r_2} \cup \cdots \cup \{P(x,r_1,\ldots,r_k)\}_{r_1,\ldots,r_k}$.
Soundness: any proof in the above format corresponds to an "unrolled" IP prover. ∎

In sum: $\mathsf{PSPACE} \subseteq \mathsf{PCP} \subseteq \mathsf{NEXP}$. We will see that $\mathsf{PCP} = \mathsf{NEXP}$ 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.