CS 6810: Theory of Computing **Fall 2021**

Lectures 21 & 22: November 9 & 11, 2021

Lecturer: Eshan Chattopadhyay *Scribe: Jesse Goodman*

In which we further study the connection between PCPs and hardness of approximation, and introduce the BLR linearity test.

1 Recap and overview

Probabilistically checkable proofs and the PCP theorem In the previous lecture, we introduced *probabilistically checkable proofs* (PCPs), a new type of (non-interactive) proof system which has surprising connections to: (1) the complexity class NP; and (2) hardness of approximation. In this lecture, we revisit the definition of PCPs and further explore these connections.

We start by recalling the verifier-based definition of the complexity class NP, which says that a language $L \subseteq \{0,1\}^*$ is in NP if membership in L can be efficiently and deterministically *verified*. One way to formalize this is to say that $L \in \mathsf{NP}$ if there exists a (deterministic) poly-time machine V such that for any $x \in \{0,1\}^*$, the following holds.

- **Completeness**: $x \in L \implies$ there exists a "proof" $\pi \in \{0,1\}^*$ such that $V^\pi(x) = 1$.
- **Soundness**: $x \notin L \implies$ for all "proofs" $\pi \in \{0,1\}^*$ it holds that $V^\pi(x) = 0$.

Above, recall that V^π denotes the machine V with oracle/query access (i.e., random access) to π.

The complexity class PCP, on the other hand, is simply the collection of all languages $L \subseteq \{0,1\}^*$ for which membership in L can be efficiently and *probabilistically* verified. The verifier V is now allowed to use randomness, and we simply require that completeness and soundness hold with sufficiently high probability (over the randomness of V).

We are now ready to recall the formal definition of the complexity class PCP. Beyond the above requirements, it will be useful to quantify the exact number of random bits r used by the verifier V, in addition to the number of queries q made by the verifier V to the proof π. We will eventually see that, depending on the settings of these parameters, PCPs will provide surprising new characterizations of several complexity classes.

Definition 1.1 (Complexity class $\mathsf{PCP}_{c,s}(r,q)$)**.** *For any $r, q : \mathbb{N} \to \mathbb{N}$ and $0 \le s \le c \le 1$, we define the class $\mathsf{PCP}_{c,s}(r,q)$ as follows. For any language $L \subseteq \{0,1\}^*$, we say that $L \in \mathsf{PCP}_{c,s}(r,q)$ iff there exists a probabilistic poly-time verifier V such that for any $x \in \{0,1\}^*$, the following holds.*

- **Completeness**: $x \in L \implies$ *there exists some $\pi \in \{0,1\}^*$ such that* $\Pr[V^\pi(x) = 1] \ge c$.
- **Soundness**: $x \notin L \implies$ *for all $\pi \in \{0,1\}^*$ it holds that* $\Pr[V^\pi(x) = 1] \le s$.
- **Efficiency**: *V uses at most $r(|x|)$ bits of randomness and makes at most $q(|x|)$ queries to π.*

Remark 1.2. *For convenience, we use the shorthand $\mathsf{PCP}(r,q) := \mathsf{PCP}_{1,1/2}(r,q)$. Furthermore, we will occasionally abuse notation and write $\mathsf{PCP}(\mathcal{R},\mathcal{Q})$ for some function families $\mathcal{R},\mathcal{Q}$; we define this class as $\mathsf{PCP}(\mathcal{R},\mathcal{Q}) := \bigcup_{r\in\mathcal{R},q\in\mathcal{Q}} \mathsf{PCP}(r,q)$.*

Given the above definitions, it is immediate that $\mathsf{NP} = \mathsf{PCP}_{1,0}(0, \mathsf{poly}(n))$. Suprisingly, however, it turns out that by providing the PCP verifier with just a few bits of randomness (and slightly relaxing the soundness requirement), PCPs offer another characterization of NP in which the PCP verifier only needs to read a *constant number* of bits from the proof. This landmark result is known as the *PCP Theorem*, and was proven by Arora and Safra [AS98] and Arora, Lund, Motwani, Sudan, and Szegedy [ALM$^+$98].

Theorem 1.3 (The PCP Theorem [AS98, ALM$^+$98])**.**

$$\mathsf{NP} = \mathsf{PCP}(O(\log n), O(1)).$$

It is not difficult to show[1] that $\mathsf{NP} \supseteq \mathsf{PCP}(O(\log n), O(1))$, and thus the *PCP theorem* often simply refers to the inclusion $\mathsf{NP} \subseteq \mathsf{PCP}(O(\log n), O(1))$. Indeed, it is also this direction of the theorem that yields the following fascinating philosophical takeaway: *if a given mathematical statement admits a proof that can be efficiently and deterministically verified, then it also admits a proof that can be efficiently verified by checking just a constant number of (random) locations in the proof.*[2]

Overview Beyond the above fascinating philosophical takeaway, the PCP theorem has surprising applications in hardness of approximation. We will start by exploring these applications in Section 2. Then, in Section 3 we move towards proving (a weaker version of) the PCP theorem. A key tool in this proof is the *BLR linearity test*, which is an efficient algorithm for testing whether a function is linear. We introduce this tool and prove its correctness in Section 3, and we will use it to prove the weak PCP theorem (Theorem 3.1) in the next lecture.[3]

2 The connection between PCPs and hardness of approximation

As it turns out, the PCP theorem has important applications in *hardness of approximation.* In fact, the PCP theorem is actually equivalent to a statement about the hardness of approximating the solution to a specific problem. In this section, we formally describe and prove this equivalence.

Constraint satisfaction problems The exact problem that will witness this equivalence is a generalization of the classical $\mathsf{3SAT}$ problem, in which (1) the clauses may be replaced with arbitrary functions that depend on a limited number of variables; and (2) we become interested in the *maximum fraction* of clauses that can be satisfied, not just whether *all of them* can be satisfied. To make things more formal, we introduce general boolean formula known as q-Constraint Satisfaction Problems ($q\mathsf{CSP}$s).

Definition 2.1 ($q\mathsf{CSP}$)**.** *A $q\mathsf{CSP}$ over n variables with m clauses is a sequence $\phi = (\phi_1, \ldots, \phi_m)$, where each clause $\phi_i : \{0,1\}^n \to \{0,1\}$ is a boolean function that depends on at most q of its input bits. For every $x \in \{0,1\}^n$ we define $\mathsf{val}_x(\phi) := \frac{1}{m}\sum_{i\in[m]} \phi_i(x)$ to be the number of clauses satisfied by x. We let $\mathsf{val}(\phi) := \max_x \mathsf{val}_x(\phi)$, and we say that ϕ is satisfiable if $\mathsf{val}(\phi) = 1$.*

[1]Indeed, the more general result $\mathsf{PCP}(r, q) \subseteq \mathsf{NTIME}(2^{O(r)} \cdot q \cdot n^{O(1)})$ can be proven as follows. First, observe that the proofs read by the PCP verifier may be assumed to have length at most $2^r q$, since the PCP verifier checks at most q locations of the proof for each of the 2^r possible fixings of its randomness. Then, have the NTIME verifier simulate the PCP verifier on all 2^r possible fixings of its randomness and accept iff the PCP verifier always accepts.

[2]Here, we are referring to the language L which contains all strings $x \in \{0,1\}^*$ that are mathematical statements which have polynomial sized proofs written in a way that can be formally verified by a single poly-time verifier (e.g., a human). Clearly $L \in \mathsf{NP}$, and the PCP theorem thus implies $L \in \mathsf{PCP}(O(\log n), O(1))$.

[3]Actually, this will happen in Lecture 24, as the next lecture will be a guest lecture on a different topic.

Remark 2.2. *Observe that we may always assume ϕ is encoded using $\mathsf{poly}(n, m, 2^q)$ bits, since each ϕ_i can be written as a boolean function $\phi_i : \{0,1\}^q \to \{0,1\}$, and there are 2^{2^q} such functions.*

We will see that the PCP theorem is equivalent to a theorem which roughly asserts that $\mathsf{val}(\phi)$ of a $q\mathsf{CSP}$ ϕ cannot even be approximated (unless $\mathsf{P} = \mathsf{NP}$). Towards this end, we must first review the notions of *approximation algorithms* and *promise problems.*

We start by defining $\mathsf{MAX}\text{-}q\mathsf{CSP}$ as the optimization problem of determining $\mathsf{val}(\phi)$ of a $q\mathsf{CSP}$ ϕ. Then, for any $0 \leq \rho \leq 1$, recall that $\mathcal{A}$ is a *ρ-approximation algorithm* for $\mathsf{MAX}\text{-}q\mathsf{CSP}$ if $\mathcal{A}$ runs in poly-time and

$$\rho \cdot \mathsf{val}(\phi) \leq \mathcal{A}(\phi) \leq \mathsf{val}(\phi)$$

for any $q\mathsf{CSP}$ ϕ.

We now need a formal way to assert the non-existence of certain approximation algorithms for $\mathsf{MAX}\text{-}q\mathsf{CSP}$. For this, we need the notion of *promise problems.* Recall that a promise problem is simply a tuple $L = (L_{\text{Yes}}, L_{\text{No}})$ of disjoint sets $L_{\text{Yes}}, L_{\text{No}} \subseteq \{0,1\}^*$. We say that an algorithm *decides* the promise problem if it accepts all inputs $x \in L_{\text{Yes}}$ and rejects all inputs $x \in L_{\text{No}}$. On the other hand, for inputs $x \notin L_{\text{Yes}} \cup L_{\text{No}}$, the algorithm is allowed to reject *or* accept (as long as it halts). Promise problems naturally generalize decision problems, which require that $L_{\text{Yes}} \cup L_{\text{No}} = \{0,1\}^*$. We will be interested in the following problem.

Definition 2.3 ($\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$)**.** *The promise problem $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ is the tuple $L = (L_{Yes}, L_{No})$, defined as*

$$L_{Yes} := \{\phi \in \{0,1\}^* : \phi \textit{ is a } q\mathsf{CSP} \textit{ with } \mathsf{val}(\phi) = 1\}$$
$$L_{No} := \{\phi \in \{0,1\}^* : \phi \textit{ is a } q\mathsf{CSP} \textit{ with } \mathsf{val}(\phi) \leq \rho\}$$

We say that the promise problem $\rho\mathsf{GAP}\text{-}q\mathsf{CSP} = (L_{\text{Yes}}, L_{\text{No}})$ is NP-hard if for any language $L \in \mathsf{NP}$, there exists a poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that the following holds for all $x \in \{0,1\}^*$: if $x \in L$ then $f(x) \in L_{\text{Yes}}$, and if $x \notin L$ then $f(x) \in L_{\text{No}}$. We now proceed by showing the connection between the (non)existence of approximation algorithms for $\mathsf{MAX}\text{-}q\mathsf{CSP}$ and the NP-hardness of $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$, and then show the equivalence of the PCP theorem to the NP-hardness of $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$. As a result, we obtain a deep connection between the PCP theorem and hardness of approximation.

Claim 2.4. *If $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ is NP-hard, then for all $\rho' > \rho$, there cannot exist a ρ'-approximation algorithm for $\mathsf{MAX}\text{-}q\mathsf{CSP}$ (unless $\mathsf{P} = \mathsf{NP}$).*

Proof. Assume that the promise problem $\rho\mathsf{GAP}\text{-}q\mathsf{CSP} = (L_{\text{Yes}}, L_{\text{No}})$ is NP-hard and that there exists a ρ'-approximation algorithm $\mathcal{A}$ for $\mathsf{MAX}\text{-}q\mathsf{CSP}$, for some $\rho' > \rho$. To prove the claim, it suffices to show that this implies $\mathsf{P} = \mathsf{NP}$.

Fix any language $L' \in \mathsf{NP}$, and note that by the NP-hardness of $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ there exists a poly-time computable function f such that $x \in L' \implies f(x) \in L_{\text{Yes}}$ and $x \notin L' \implies f(x) \in L_{\text{No}}$. Now, consider an algorithm $\mathcal{B}$ which takes its input $x \in \{0,1\}^*$ and outputs 1 iff $\mathcal{A}(f(x)) > \rho$. Notice that $\mathcal{B}$ clearly runs in poly-time, and furthermore that it decides L'. To see why the latter holds, simply observe that if $x \in L'$ then $f(x) \in L_{\text{Yes}}$ and thus $\mathsf{val}(f(x)) = 1$. By definition of $\mathcal{A}$, we get that $\mathcal{A}(f(x)) \geq \rho' \cdot 1 > \rho$, and thus $\mathcal{B}$ accepts. On the other hand, if $x \notin L'$ then $f(x) \in L_{\text{No}}$ and thus $\mathsf{val}(f(x)) \leq \rho$. By definition of $\mathcal{A}$, we get that $\mathcal{A}(f(x)) \leq \rho$ and thus $\mathcal{B}$ rejects. Thus $L' \in \mathsf{P}$, which implies that $\mathsf{NP} \subseteq \mathsf{P}$ and thus $\mathsf{P} = \mathsf{NP}$. □

The equivalence Finally, we are ready to state and prove the equivalence between the PCP theorem and hardness of approximation.

Theorem 2.5. *The following two statements are equivalent.*

1. **The PCP theorem**: *There exist constants* $0 \leq \rho < 1$ *and* $q \in \mathbb{N}$ *such that*
$$\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q).$$
2. **Hardness of approximation**: *There exist constants* $0 \leq \rho < 1$ *and* $q \in \mathbb{N}$ *such that*
$$\rho\mathsf{GAP}\text{-}q\mathsf{CSP} \textit{ is } \mathsf{NP}\textit{-hard.}$$

Before we prove this equivalence, we must make some remarks. First, we point out that the PCP theorem written in Theorem 2.5 is slightly different than the PCP theorem written in Theorem 1.3. However, it is not difficult to show that these two PCP theorems are equivalent.[4] Next, we remark that while we will prove the Equivalence Theorem 2.5, the main challenge is actually proving that one of the statements listed in Theorem 2.5 is true. Finally, we emphasize the hardness of approximation results obtained through combining this equivalence theorem with the PCP theorem. In particular, note that by combining Theorem 2.5 with Theorem 1.3 and Claim 2.4, we obtain the following corollary.

Corollary 2.6. *There exist constants* $0 \leq \rho < 1$ *and* $q \in \mathbb{N}$ *such that the following holds. There does not exist a* ρ*-approximation algorithm for* $\mathsf{MAX}\text{-}q\mathsf{CSP}$ *(unless* $\mathsf{P} = \mathsf{NP}$*).*

Thus, we have finally seen how the PCP theorem yields hardness of approximation results against a specific problem: $\mathsf{MAX}\text{-}q\mathsf{CSP}$. To obtain hardness of approximation results against another problem $\mathcal{P}$, all that is needed is a so-called *approximation-preserving reduction* from $\mathsf{MAX}\text{-}q\mathsf{CSP}$ to $\mathcal{P}$. Following the proof of the PCP theorem, a collection of papers have developed a rich theory around such reductions, and have subsequently proved near-optimal hardness of approximation results for several classical problems in computer science.

Finally, we conclude this section by proving Theorem 2.5 via the following two claims.

Claim 2.7. *Suppose there exist constants* $0 \leq \rho < 1$ *and* $q \in \mathbb{N}$ *such that* $\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q)$*. Then it follows that* $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ *is* NP*-hard for the same constants* ρ, q*. In short,*
$$\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q) \implies \rho\mathsf{GAP}\text{-}q\mathsf{CSP} \textit{ is } \mathsf{NP}\textit{-hard.}$$

Claim 2.8. *Suppose that there exist constants* $0 \leq \rho < 1$ *and* $q \in \mathbb{N}$ *such that* $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ *is* NP*-hard. Then it follows that* $\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q)$ *for the same constants* ρ, q*. In short,*
$$\rho\mathsf{GAP}\text{-}q\mathsf{CSP} \textit{ is } \mathsf{NP}\textit{-hard.} \implies \mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q).$$

Clearly, Theorem 2.5 follows immediately from Claim 2.7 and Claim 2.8. In fact, these claims prove something slightly stronger: that the constants ρ, q appearing in the first item of Theorem 2.5 are identical to those appearing in the second item. This draws equivalences between (1) the *soundness* ρ of the PCP system and the *approximation gap* ρ of the promise problem $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$; and (2) the *query complexity* q of the PCP system and the *arity* q of $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$, i.e., the number of variables that each clause depends on.

We now prove Claim 2.7.

[4]In particular, note that we can always do error reduction on the soundness parameter ρ if it is greater than $1/2$; and we can fix the query parameter $O(1)$ to a universal constant q by reducing each NP language to $\mathsf{3SAT}$ before constructing a PCP verifier.

Proof of Claim 2.7. Assume that there exist constants $0 \le \rho < 1$ and $q \in \mathbb{N}$ such that $\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q)$. We wish to show that $\rho\mathsf{GAP}\text{-}q\mathsf{CSP} = (L_{\text{Yes}}, L_{\text{No}})$ is NP-hard: that is, we must show that for every language $L' \in \mathsf{NP}$, there exists a poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $x \in L' \implies f(x) \in L_{\text{Yes}}$ and $x \notin L' \implies f(x) \in L_{\text{No}}$.

Towards this end, fix any $L' \in \mathsf{NP}$. Since $\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q)$, there must be some constant C such that $L' \in \mathsf{PCP}_{1,\rho}(C \log n, q)$. And by definition of PCP, we know that there is a (probabilistic) poly-time verifier V such that for all $x \in \{0,1\}^*$, the following hold.

- **Completeness**: $x \in L' \implies \exists \pi \in \{0,1\}^*$ such that $\Pr[V^\pi(x) = 1] = 1$.
- **Soundness**: $x \notin L' \implies \forall \pi \in \{0,1\}^*$ it holds that $\Pr[V^\pi(x) = 1] \le \rho$.
- **Efficiency**: V uses at most $C \log |x|$ bits of randomness and makes at most q queries to π.

Without loss of generality, we may assume the proofs π queried by the verifier V have length exactly $L := 2^{C \log |x|} q$, since V checks at most q locations of the proof for each of the $2^{C \log |x|}$ possible fixings of its randomness. Furthermore, notice that for any input $x \in \{0,1\}^*$ and any fixing $\widetilde{r} \in \{0,1\}^{C \log |x|}$ of the verifier's randomness, the verifier becomes a deterministic function becomes a deterministic function of at most q symbols in the proof π. Let $\phi_{x,\widetilde{r}} : \{0,1\}^L \to \{0,1\}$ denote this function, which depends on $\le q$ of its inputs, and let $\phi_x = (\phi_{x,\widetilde{r}})_{\widetilde{r} \in \{0,1\}^{C \log |x|}}$ denote the collection of these functions over all $\widetilde{r}$. Clearly, ϕ_x is a $q\mathsf{CSP}$. Furthermore, by the completeness and soundness of the PCP verifier V, we know:

$$\begin{aligned} &x \in L' \implies \exists \pi \text{ such that } \Pr[V^\pi(x) = 1] = 1 \implies \mathsf{val}(\phi_x) = 1 \\ &x \notin L' \implies \forall \pi \text{ it holds that } \Pr[V^\pi(x) = 1] \le \rho \implies \mathsf{val}(\phi_x) \le \rho. \end{aligned}$$

We are now ready to construct our poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$. Given as input any $x \in \{0,1\}^*$, our function f iterates over all $\widetilde{r} \in \{0,1\}^{C \log |x|}$, simulating the verifier V on fixed $x, \widetilde{r}$ to collect the functions $\phi_{x,\widetilde{r}}$, which are over the (unfixed) proof symbols. Then, the function f outputs $\phi_x = (\phi_{x,\widetilde{r}})_{\widetilde{r} \in \{0,1\}^{C \log |x|}}$. Since the verifier V runs in $\mathsf{poly}(|x|)$ time and the proof has length $L = \mathsf{poly}(|x|)$, each function $\phi_{x,\widetilde{r}}$ can be built in $\mathsf{poly}(|x|)$ time via a Cook-Levin-type argument. And since there are $2^{C \log |x|} = \mathsf{poly}(|x|)$ such functions $\phi_{x,\widetilde{r}}$ in ϕ_x, we see that ϕ_x can be built in $\mathsf{poly}(|x|)$ time and thus f is a poly-time computable function.

Now, recall that we showed above that $x \in L' \implies \mathsf{val}(\phi_x) = 1$. Since $f(x) = \phi_x$, we get that $\mathsf{val}(f(x)) = 1$. By definition of $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ (Definition 2.3), this implies that $f(x) \in L_{\text{Yes}}$. Similarly, we have $x \notin L' \implies \mathsf{val}(\phi_x) \le \rho \implies \mathsf{val}(f(x)) \le \rho \implies f(x) \in L_{\text{No}}$. Thus, we have proven the efficiency and correctness requirements of f, thereby showing it is a poly-time reduction from L' to $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$. Since we picked an arbitrary $L' \in \mathsf{NP}$, we get that $\rho\mathsf{GAP}\text{-}q\mathsf{CSP}$ is NP-hard, as desired. $\square$

Finally, we conclude our proof of Theorem 2.5 by proving Claim 2.8.

Proof of Claim 2.8. Assume that there are constants $0 \le \rho < 1$ and $q \in \mathbb{N}$ such that $\rho\mathsf{GAP}\text{-}q\mathsf{CSP} = (L_{\text{Yes}}, L_{\text{No}})$ is NP-hard. In other words, for every $L' \in \mathsf{NP}$ there is a poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $x \in L' \implies f(x) \in L_{\text{Yes}}$ and $x \notin L' \implies f(x) \in L_{\text{No}}$. We wish to show that $\mathsf{NP} = \mathsf{PCP}_{1,\rho}(O(\log n), q)$. As mentioned earlier in these notes, it is easy to show $\mathsf{PCP}_{1,\rho}(O(\log n), q) \subseteq \mathsf{NP}$ for constant q. Thus, all that remains is to prove $\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}(O(\log n), q)$.

Fix any $L' \in \mathsf{NP}$, and let $f : \{0,1\}^* \to \{0,1\}^*$ be the poly-time computable function with $x \in L' \implies f(x) \in L_{\text{Yes}}$ and $x \notin L' \implies f(x) \in L_{\text{No}}$. We now construct a PCP verifier $\widetilde{V}$ as follows. On input $x \in \{0,1\}^*$, it first computes the $q\mathsf{CSP}$ $\phi_x := f(x)$. Suppose ϕ_x is over t variables and m clauses $\phi_{x,1}, \dots, \phi_{x,m}$. The PCP verifier $\widetilde{V}$ then picks a (uniformly) random index $i \sim [m]$, examines the clause $\phi_{x,i}$, and records the $\leq q$ input coordinates on which it depends.[5] Finally, the verifier queries its proof π at locations $\pi_{j_1}, \dots, \pi_{j_q}$ and outputs $\phi_{x,i}(\pi_{j_1}, \dots, \pi_{j_q})$.[6]

We claim that $\widetilde{V}$ is an efficient, complete, and sound PCP verifier for L', and thus $L' \in \mathsf{PCP}_{1,\rho}(O(\log n), q)$. To see why, we first observe that $\widetilde{V}$ runs in (probabilistic) polynomial time, since f runs in poly-time. Furthermore, the latter also implies that $m = \mathsf{poly}(|x|)$, and thus the verifier $\widetilde{V}$ uses $\log m = \log \mathsf{poly}(|x|) = O(\log(|x|))$ bits of randomness. Since the verifier clearly queries the proof at $\leq q$ locations, we conclude that our verifier $\widetilde{V}$ is efficient.

To see why $\widetilde{V}$ has completeness, consider any $x \in L'$ and the $q\mathsf{CSP}$ $\phi_x := f(x)$ over (say) t variables. By definition of f, we know that $\phi_x \in L_{\text{Yes}}$, which means $\mathsf{val}(\phi_x) = 1$. Thus there must be some assignment $\pi \in \{0,1\}^t$ such that $\phi_{x,i}(\pi) = 1$ for every clause $\phi_{x,i}$ in ϕ_x. Give such an assignment π as an oracle, it is clear that $\widetilde{V}^\pi(x)$ will output 1 with probability 1, since π satisfies each clause in ϕ_x. Thus

$$x \in L' \implies \exists \pi \in \{0,1\}^* \text{ such that } \Pr[V^\pi(x) = 1] = 1,$$

and completeness holds.

To see why $\widetilde{V}$ has soundness, consider any $x \notin L'$ and the $q\mathsf{CSP}$ $\phi_x := f(x)$ over (say) t variables. By definition of f, we know that $\phi_x \in L_{\text{No}}$, which means $\mathsf{val}(\phi_x) \leq \rho$. In other words, for *any* assignment $\pi \in \{0,1\}^t$ it holds that $\phi_{x,i}(\pi) = 1$ for at most ρ fraction of the clauses $\phi_{x,i}$ in ϕ_x. Thus for any assignment π as an oracle, it is clear that $\widetilde{V}^\pi(x)$ will output 1 with probability at most ρ, since this is the maximum probability that a randomly selected clause in ϕ_x is satisfied by π. Thus

$$x \notin L' \implies \forall \pi \in \{0,1\}^* \text{ it holds that } \Pr[V^\pi(x) = 1] \leq \rho,$$

and soundness holds.

Thus, we conclude that $\widetilde{V}$ is an efficient, complete, and sound PCP verifier for L', and thus $L' \in \mathsf{PCP}_{1,\rho}(O(\log n), q)$. Since we picked an arbitrary $L' \in \mathsf{NP}$, we get that $\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}(O(\log n), q)$, as desired. $\square$

3 The weak PCP theorem and the BLR linearity test

The weak PCP theorem In the previous section, we saw that PCPs have a deep connection to hardness of approximation. Given this connection, we would now like to actually prove the PCP theorem (Theorem 1.3), so that we might unlock the hardness of approximation applications implied by Theorem 2.5 and Corollary 2.6. Due to time constraints, however, we will not be able to cover the full proof of the PCP theorem. Instead, we set out to prove the following weaker version.

Theorem 3.1 (The Weak PCP Theorem)**.**

$$\mathsf{NP} \subseteq \mathsf{PCP}(\mathsf{poly}(n), O(1)).$$

While the weak PCP theorem will not be strong enough to yield applications in hardness of approximation, its proof is enlightening and contains some similar ingredients to the proof of the

[5]We assume the definition of $q\mathsf{CSP}$ has the encoding of such a formula include this information in each clause.

[6]Technically, the latter is abuse of notation, as our definition of $q\mathsf{CSP}$ had each clause as a deterministic function of all the t variables (instead of just q), and just depend on q of them. We can easily rectify this by plugging in arbitrary bits into the input locations on which $\phi_{x,i}$ does not depend.

actual PCP theorem. One such ingredient is the *BLR linearity test.* In the remainder of this section, we introduce this test and prove its correctness. In Lecture 24, we will show how to use this test to prove Theorem 3.1.

Property testing Before we discuss the BLR linearity test, we must introduce the field of *property testing.* In this area, one studies the question of whether some property of an unknown function $f : \{0,1\}^n \to \{0,1\}$ can be determined using just a few queries into that function. To make things more formal, if we let $\mathcal{F}$ denote the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}$, then a *property* $\mathcal{P}$ is simply a subset $\mathcal{P} \subseteq \mathcal{F}$. At a high level, we would like to construct an efficient algorithm $\mathcal{A}$ that, given oracle access to an unknown function f, can efficiently check whether f has property $\mathcal{P}$. Here, efficiency is measured by the number of queries to f made by $\mathcal{A}$, and we are not concerned about the runtime of $\mathcal{A}$.

Clearly, we can always construct an algorithm $\mathcal{A}$ that determines whether $f \in \mathcal{P}$ by making 2^n queries to f (i.e., at each of its input points). On the other hand, it is not difficult to construct simple properties for which 2^n queries are necessary: for example, take any property $\mathcal{P}$ for which membership $f \in \mathcal{P}$ depends on its evaluation for every input x. To cope with this, we make two relaxations: (1) we allow the tester $\mathcal{A}$ to use unlimited randomness; and (2) we only require the tester $\mathcal{A}$ to succeed with sufficiently high probability. Furthermore, we tailor our definition of "sufficiently high probability" to allow the tester $\mathcal{A}$ to fail if f doesn't have property $\mathcal{P}$, but is extremely close to having it.

Given this intuition, we are almost ready to formally define a property tester $\mathcal{A}$. First, we just need to formalize the notion of "closeness" of a function f to a property $\mathcal{F}$. Given functions $f, g : \{0,1\}^n \to \{0,1\}$, we define the distance $\mathsf{dist}(f,g)$ between them as the normalized Hamming distance between their truth tables:

$$\mathsf{dist}(f,g) := \Pr_{x \sim \{0,1\}^n}[f(x) \neq g(x)].$$

That is, $\mathsf{dist}(f,g)$ counts the fraction of inputs on which f, g differ. Next, we define the distance of f from a property $\mathcal{P}$ as its distance to the closest $g \in \mathcal{P}$:

$$\mathsf{dist}(f,\mathcal{P}) := \min_{g \in \mathcal{P}} \mathsf{dist}(f,g).$$

We are now ready to formally define a property tester.

Definition 3.2 (Property tester)**.** *Fix any functions $q : \mathbb{N} \to \mathbb{N}$ and $\lambda : \mathbb{N} \to \mathbb{R}^+$, and any property $\mathcal{P}$. We say that a (randomized) algorithm $\mathcal{A}$ is a (q,λ)-*tester *for $\mathcal{P}$ if for every function $f : \{0,1\}^n \to \{0,1\}$, all of the following hold:*

- **Completeness**: $f \in \mathcal{P} \implies \Pr[\mathcal{A}^f = 1] = 1$.
- **Soundness**: $f \notin \mathcal{P} \implies \Pr[\mathcal{A}^f = 0] \geq \lambda(n) \cdot \mathsf{dist}(f,\mathcal{P})$.
- **Efficiency**: *$\mathcal{A}$ makes at most $q(n)$ queries to f.*

Property testing is a beautiful subfield of theoretical computer science, and many results in the field rely on cool structural results from combinatorics, such as the *graph removal lemma* and *Szemerédi's regularity lemma.* Typically, one hopes to construct good tests for *global* properties of boolean functions, using very little *local* information (i.e., one hopes to construct (q,λ)-testers where q, λ are constants). For a detailed introduction to property testing, we refer the reader to the excellent book of Goldreich [Gol17].

The BLR linearity test Now that we have given a brief introduction to property testing, we are ready to define the BLR linearity test. This property tester, introduced by Blum, Luby, and Rubinfeld [BLR93], will be crucial to our proof of the weak PCP theorem. To formally introduce the BLR test, we must first define what it means for a function $f : \{0,1\}^n \to \{0,1\}$ to be *linear*. Here, we define f to be linear if for all $x, y \in \{0,1\}^n$ it holds that $f(x) + f(y) = f(x+y)$, where $+$ refers to addition modulo 2 (i.e., over $\mathbb{F}_2$). We define LIN to be the collection of all linear functions $f : \{0,1\}^n \to \{0,1\}$, and introduce the BLR test, below.

Theorem 3.3 (BLR test [BLR93])**.** *There exists a* $(3,1)$*-tester* BLR *for* LIN*. Given oracle access to a function* $f : \{0,1\}^n \to \{0,1\}$*, the tester* BLR *simply draws uniform* $x, y \sim \{0,1\}^n$ *and accepts iff* $f(x) + f(y) = f(x+y)$*.*

The BLR test is remarkably simple, yet it is able to test for the global property of *linearity* using extremely few (local) queries $q = 3$, while simultaneously having excellent soundness $\lambda = 1$. We conclude this section by proving the following slightly weaker version of Theorem 3.3.

Theorem 3.4. BLR *is a* $(3, 2/9)$*-tester for* LIN*.*

While the proof of Theorem 3.3 is not too long, it goes by way of *Fourier analysis* (of Boolean functions), which we do not have ample time to introduce.[7] On the other hand, Theorem 3.4 can be proven directly, without using any additional machinery. We present it now.

The key tool we use to prove Theorem 3.4 is an auxiliary function $g : \{0,1\}^n \to \{0,1\}$ we call the *majority decoder*. Intuitively, given a function f to be tested, g can be thought of as a "smoothening" of f that makes it "more linear." In more detail, recall that f is linear iff for any fixed $x \in \{0,1\}^n$ and all $y \in \{0,1\}^n$ it holds that $f(y) + f(x+y) = f(x)$. The majority decoder g will be defined so that $g(x)$ takes the majority value of $f(y) + f(x+y)$ over all y. More formally, it is defined as follows.

$$g(x) := \begin{cases} 1 & \text{if } \Pr_{y \sim \{0,1\}^n}[f(y) + f(x+y) = 1] \geq 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

Using the majority decoder, we present two simple claims from which Theorem 3.4 will easily follow.

Claim 3.5. *For any function* $f : \{0,1\}^n \to \{0,1\}$*, it holds that* $\Pr[\mathsf{BLR}^f \text{ rejects}] \geq \frac{1}{2} \cdot \mathsf{dist}(f, g)$*.*

Claim 3.6. *For any function* $f : \{0,1\}^n \to \{0,1\}$*, if* $\Pr[\mathsf{BLR}^f \text{ rejects}] < \frac{2}{9}$ *then* $g \in \mathsf{LIN}$*.*

Before proving these two claims, we show how they immediately imply Theorem 3.4.

Proof of Theorem 3.4. As per Definition 3.2, we must show that BLR is complete, sound, and efficient for testing the class LIN of linear functions. Efficiency is straightforward, since the tester BLR clearly just makes $q = 3$ queries. The completeness is also straightforward, since the definition of linearity and the BLR test implies that BLR will always accept if f is linear. Thus, all that remains is to show that BLR is sound: that is, $f \notin \mathsf{LIN} \implies \Pr[\mathsf{BLR}^f = 0] \geq \frac{2}{9} \cdot \mathsf{dist}(f, \mathsf{LIN})$.

To show that this is true, we proceed with two simple cases: either g is linear, or g is not. If g is linear, then $\mathsf{dist}(f, g) \geq \mathsf{dist}(f, \mathsf{LIN})$, by definition of $\mathsf{dist}(\cdot, \cdot)$. Thus by Claim 3.5 we get

$$\Pr[\mathsf{BLR}^f \text{ rejects}] \geq \frac{1}{2} \cdot \mathsf{dist}(f, g) \geq \frac{1}{2} \cdot \mathsf{dist}(f, \mathsf{LIN}) \geq \frac{2}{9} \cdot \mathsf{dist}(f, \mathsf{LIN}).$$

[7]For an introduction to this area and a proof of Theorem 3.3, we recommend the excellent book of O'Donnell [O'D14].

On the other hand, if g is not linear, then by Claim 3.6 we get $\Pr[\mathsf{BLR}^f \text{ rejects}] \geq \frac{2}{9}$. Since $\mathsf{dist}(\cdot,\cdot)$ is a probability by its definition, we know it takes values in $[0,1]$, and in particular is at most 1. Thus we again have

$$\Pr[\mathsf{BLR}^f \text{ rejects}] \geq \frac{2}{9} \geq \frac{2}{9} \cdot \mathsf{dist}(f, \mathsf{LIN}). \qquad \square$$

Thus, all that remains is to prove Claim 3.5 and Claim 3.6. We start with the former.

Proof of Claim 3.5. For every $x \in \{0,1\}^n$, we define a measure $a(x)$ which counts the fraction of $y \in \{0,1\}^n$ with which x does not "behave linearly" under f. More formally, we define

$$a(x) := \Pr_{y \sim \{0,1\}^n}[f(y) + f(x+y) \neq f(x)].$$

Notice that $a(x) > 1/2$ if and only if x does not behave linearly with most y. More formally, using the definitions of this measure a and our majority decoder g, it is straightforward to verify that

$$a(x) > 1/2 \iff f(x) \neq g(x).$$

Using this observation, we are motivated to define a set of "bad" inputs on which f disagrees with the majority decoder, g:

$$\mathsf{BAD} := \{x \in \{0,1\}^n : f(x) \neq g(x)\} = \{x \in \{0,1\}^n : a(x) > 1/2\}.$$

It is now a straightforward calculation to prove the claim:

$$\begin{aligned}
\Pr[\mathsf{BLR}^f \text{ rejects}] &:= \Pr_{x,y}[f(x) + f(y) \neq f(x+y)] \\
&= \Pr_{x,y}[f(x) \neq f(y) + f(x+y)] \\
&= \mathbb{E}_x[a(x)] \\
&\geq \frac{1}{2} \cdot \Pr_x[x \in \mathsf{BAD}] \\
&= \frac{1}{2} \cdot \Pr_x[f(x) \neq g(x)] \\
&= \frac{1}{2} \cdot \mathsf{dist}(f, g).
\end{aligned}$$

$\square$

We now conclude our proof of Theorem 3.4 by proving Claim 3.6. In order to prove this result, we will introduce another measure $\lambda(x)$ which counts the fraction of $y \in \{0,1\}^n$ on which g "fails to make x behave linearly with y under f." More formally, we define

$$\lambda(x) := \Pr_{y \sim \{0,1\}^n}[f(y) + f(x+y) \neq g(x)].$$

Now, in order to prove Claim 3.6, we will use the following subclaim.

Subclaim 3.7.

$$\Pr[\mathsf{BLR}^f \textit{ rejects}] < \frac{2}{9} \implies \lambda(x) < \frac{1}{3} \textit{ for all } x \in \{0,1\}^n.$$

We now proceed by showing how Subclaim 3.7 implies Claim 3.6. Then, we will finally conclude with the proof of Subclaim 3.7.

Proof of Claim 3.6. We wish to show that $\Pr[\mathsf{BLR}^f \text{ rejects}] < \frac{2}{9} \implies g \in \mathsf{LIN}$. Towards this end, for each $x \in \{0,1\}^n$ we define a set of "bad" y for which g "fails to make x behave linearly with y under f." That is, we define

$$\mathsf{BAD}_x := \{y \in \{0,1\}^n : g(x) \neq f(y) + f(x+y)\}.$$

By Subclaim 3.7, we know that $|\mathsf{BAD}_x| < 2^n/3$ for all x. The goal now is to use this to show $g \in \mathsf{LIN}$. That is, we must show that for any $v_1, v_2 \in \{0,1\}^n$ it holds that $g(v_1)+g(v_2) = g(v_1+v_2)$. Towards this end, we observe the following:

- For any $y \notin \mathsf{BAD}_{v_1}$, it holds that $g(v_1) = f(y) + f(y+v_1)$.
- For any $y \notin \mathsf{BAD}_{v_2}$, it holds that $g(v_2) = f(y) + f(y+v_2)$.
- For any $y \notin \mathsf{BAD}_{v_1+v_2}$, it holds that $g(v_1+v_2) = f(y) + f(y+v_1+v_2)$.

As it turns out, it will be more handy to re-write the last bullet in another way. First, define the set $\mathsf{BAD}^*_{v_1+v_2} := \{y : y + v_1 \in \mathsf{BAD}_{v_1+v_2}\}$. Then, we note that $|\mathsf{BAD}^*_{v_1+v_2}| < 2^n/3$ since $|\mathsf{BAD}_{v_1+v_2}| < 2^n/3$, and we observe that we may re-write the last bullet from above as follows.

- For any $y \notin \mathsf{BAD}^*_{v_1+v_2}$, it holds that $g(v_1+v_2) = f(y+v_1) + f(y+v_2)$.

Now, since we know that $\mathsf{BAD}_{v_1}, \mathsf{BAD}_{v_2}$, and $\mathsf{BAD}^*_{v_1+v_2}$ all contain $< 2^n/3$ elements, there must exist some $z \in \{0,1\}^n$ that does not belong to any of these sets. And by our bullets above, we know that all of the following must hold:

- $g(v_1) = f(z) + f(z+v_1)$.
- $g(v_2) = f(z) + f(z+v_2)$.
- $g(v_1+v_2) = f(z+v_1) + f(z+v_2)$.

Combining all of this, we get

$$g(v_1)+g(v_2) = (f(z)+f(z+v_1)) + (f(z)+f(z+v_2)) = f(z+v_1)+f(z+v_2) = g(v_1+v_2),$$

which proves that g is linear, as desired. $\square$

At last, all that remains is to prove Subclaim 3.7. We do so below.

Proof of Subclaim 3.7. We wish to prove that $\Pr[\mathsf{BLR}^f \text{ rejects}] < \frac{2}{9} \implies \lambda(x) < \frac{1}{3}$, where $\lambda(x) := \Pr_y[f(y)+f(x+y) \neq g(x)]$. Towards this end, we will prove two bounds on the quantity

$$\Pr_{y,y'}[f(y)+f(x+y) = f(y')+f(x+y')]. \tag{1}$$

First, note that for random y, y', we have that $\Pr_{y,y'}[f(y)+f(y') = f(y+y')] = \Pr[\mathsf{BLR}^f \text{ accepts}] > 7/9$. Similarly, for any fixed x we also have $\Pr_{y,y'}[f(x+y)+f(x+y') = f(x+y+x+y')] = \Pr_{y,y'}[f(y)+f(y') = f(y+y')] = \Pr[\mathsf{BLR}^f \text{ accepts}] > 7/9$, and thus $\Pr_{y,y'}[f(x+y)+f(x+y') = f(y+y')] > 7/9$. Thus, we have

$$\begin{aligned}\Pr_{y,y'}[f(y)+f(x+y) = f(y')+f(x+y')] &= \Pr_{y,y'}[f(x+y)+f(x+y') = f(y)+f(y')] \\ &\geq \Pr_{y,y'}[f(x+y)+f(x+y') = f(y+y'), f(y)+f(y') = f(y+y')] \\ &> 5/9,\end{aligned}$$

where the last inequality follows by the union bound. Thus we have obtained a strict lower bound of 5/9 on the quantity in Equation (1). On the other hand, note that by definition of $\lambda(x)$ we have:

$$\begin{aligned}\Pr_{y,y'}[f(y)+f(x+y)=f(y')+f(x+y')] &= \Pr_{y,y'}[f(y)+f(x+y)\neq g(x), f(y')+f(x+y')\neq g(x)] \\ &\quad + \Pr_{y,y'}[f(y)+f(x+y)=g(x), f(y')+f(x+y')=g(x)] \\ &= \lambda(x)^2+(1-\lambda(x))^2.\end{aligned}$$

Thus, combining our two bounds on the quantity in Equation (1), we get

$$\lambda(x)^2+(1-\lambda(x))^2 > 5/9,$$

which can be rewritten as

$$(3\lambda(x)-1)(3\lambda(x)-2) > 0.$$

It is now straightforward to verify that in order for this to be true, it must hold that $\lambda(x) < 1/3$ or $\lambda(x) > 2/3$. However, the latter setting of $\lambda(x)$ directly contradicts the definition of the majority decoder, g. Thus it must hold that $\lambda(x) < 1/3$, as desired. □

Thus concludes the proof of Subclaim 3.7, Claim 3.6, and Theorem 3.4 (the BLR test). Next time (actually, in Lecture 24), we will see how to use the BLR test to prove the weak PCP theorem, Theorem 3.1.

References

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of computer and system sciences*, 47(3):549–595, 1993.

[Gol17] Oded Goldreich. *Introduction to property testing.* Cambridge University Press, 2017.

[O'D14] Ryan O'Donnell. *Analysis of boolean functions.* Cambridge University Press, 2014.