CMSC 39600: PCPs, codes and inapproximability Oct 9, 2007

Lecture 5: PCPs of Proximity, Proof Composition

Lecturer: Prahladh Harsha *Scribe: Bill Fefferman*

In today's lecture we will first define a new variant of PCP, called PCPs of Proximity, and then use this we use to perform "proof composition", a necessary ingredient in all existing PCP constructions to reduce the size of the alphabet or equivalently to reduce the query complexity. In fact, proof composition will be one of the phases of Dinur's proof of the PCP, that we will discuss in the next few lectures. To begin with, recall the result proved towards the end of last lectuere.

Theorem 5.1. *There exists a probabilistic verifier V, such that when V is given*

- *a circuit description C with n gates, as explicit input (that it can read in its entirety)*
- *an assignment w to C, as implicit input (that it has oracle access to)*
- *and oracle access to a proof π,*

V tosses $O(n^2)$ random coins and proceeds to query a constant, $O(1)$ locations in the assignment-proof (w,π) pair and accepts/rejects the assignment-proof pair according to the following completeness and soundness rules.

Completeness:

$$C(w) = 1 \implies \exists \pi, \Pr[V^{(w,\pi)}(C) = \mathsf{acc}] = 1.$$

Soundness: *There exists a δ_0 such that for all $\delta < \delta_0$, the following holds*

$$\exists \pi, \Pr[V^{(w,\pi)}(C) = \mathsf{acc}] \geq 1-\delta \implies \exists w', C(w') = 1 \text{ and } \delta(w,w') \leq 3\delta$$

Or equivalently

$$\delta(w, SAT(C)) > 3\delta \implies \forall \pi, \Pr[V^{(w,\pi)}(C) = \mathsf{acc}] < 1-\delta,$$

where $SAT(C)$ represents the set of satisfying assignments to the circuit C.

Throughout the notes, whenever we state two strings are "$\delta-$far" (or δ-close), we will be refering to the Hanning distance between the two strings. More precisely, the Hamming distance $\Delta(x,y)$ is the number of bits x and y agree on. x is said to be δ-close to y if $\Delta(x,y) \leq \delta n$ and δ-far otherwise. A string x is δ-close to a set A, if there exists a $y \in A$ that is δ-close to x.

5.1 PCPs of Proximity

In this section, we define the object of interest in Theorem 5.1. These objects are refered to as PCPs of proximity or assignment testers in literature.

Observe that the input to the verifier comes in two parts – the explicit input C and the implicit input w. PCPs of proximity, in general, will deal with such inputs divided into two portions, one which will be explicit, or given to the verifier outright, and the other will be only accessible as an oracle – the "implicit" input. We formalize this using the notion of "pair" language.

Definition 5.2. *A L is called a "pair language" if its instances come in two parts of the form (x, y). We will refer to the first input x as the explicit input and the second input y as the implicit input. Also for all x, define the language L_x associated with x as $L_x = \{y|(x, y) \in L\}$*

One example of a pair language, which we have already discussed in Theorem 5.1 is the following Circuit-VAL $= \{(C, w)|C(w) = 1\}$. *More generally, we will consider pair languages of the form, where the explicit input is a NP-instance and the implicit input is a candidate NP-witness.*

Definition 5.3. *Fix functions $r, q : \mathbb{Z}^{\geq 0} \to \mathbb{Z}^{\geq 0}; s, \delta : \mathbb{Z}^{\geq 0} \to [0, 1]$. We say that a pair language L has a probabilistically checkable proof of proximity (PCPP) with soundness error s and proximity parameter δ if there exists a probabilistic time verifier V, that on explicit input x and implicit input y (via oracle access) and with oracle access to a proof π, tosses at most $r(|x|)$ random coins, probes at most $q(|x|)$ locations in (y, π) and then accepts/rejects the (x, y, π) triplet meeting the following two conditions*

Completeness: *If $(x, y) \in L$, then there exists a proof π such that*

$$\Pr[V^{(y,\pi)}(x) = \mathsf{acc}] = 1 \tag{1}$$

Soundness: *If y is $\delta(|x|)$-far from L_x, then for every π, the verifier $V(x)$ accepts oracle $y \circ \pi$ with probability strictly less than $s(|x|)$. i.e.,*

$$\Pr[V^{(y,\pi)}(x) = \mathsf{acc}] < s(|x|) \tag{2}$$

We use the shorthand notation "$L \in PCPP_{1,s}[r, q, \delta]$" to state the above fact.

A couple of remarks on the definition.

Remark

- *The verifier V barely reads the second input y. Thus, it is unfair to expect it to distinguish inputs (x, y) in L from those not in L. Hence, it is only expected to distinguish between strings (x, y) which are in the language from strings (x, y) where y is δ-far from the language $L_x$language. In short, the verifier only checks if y is in the proximity of L_x. Hence, the name "PCP of Proximity".*
- *All of the parameters are measured in terms of the explicit input x*
- *A PCPP is, in some sense, both a strengthening and a weakening of a PCP. It's weaker because the PCPP only rejects those (x, y) pairs where y is δ-far from L_x. However, the PCPP is more stringent since it only has oracle access to y. It happens to be the case that the stringent condition makes it harder to build PCPPs than the correspond PCPs.*

- *There is a very well-defined sense in which PCPPs are "harder" to construct than normal PCPs. I.e., If* Circuit-VAL $\in PCPP_{1,s}[r, q, \delta]$ *then* Circuit-SAT $\in PCP_{1,s}[\pi, q]$.

Theorem 5.1 can now be restated using the above notation as follows.

Theorem 5.4. $\exists \delta_0$ *such that for all* $\delta < \delta_0$

$$\text{Circuit-VAL} \in PCPP_{1,1-\delta}[O(n^2), 17, 3\delta].$$

5.2 Proof Composition

In this section, we will introduce the notion of proof composition, an important ingredient in all PCP constructions. Before we go to proof composition, let us look at the actions of a PCP verifier more closely. On input x and oracle access to a proof π, a PCP verifier tosses some random coins and based on these random coins and the input, it queries the proof in a set number of locations and accepts/rejects depending on whether the bits of the proof in the location satisfy some Boolean predicate. Or to be more formal (or pedantic), if V is the verifier, on input x and random coins R, the verifier outputs a pair (I, D) (denoted by $V(x, R) \to (I, D)$), where $I = I(x, R)$ is the set of indices it queries the proof at and $D = D(x, r)$ the description of the Boolean predicate circuit. The verifier then proceeds to query the proof at locations I and accepts the proof if $D(\pi|_I) = 1$, that is, the proof π restricted to the index set I satisfies the Boolean predicate. The completeness and soundness in Definition 5.3 as follows:

Completeness: *If $(x, y) \in L$, then there exists a proof π such that*

$$\Pr[D((y \circ \pi)|_I) = 1] = 1 \tag{3}$$

Soundness: *If y is $\delta(|x|)$-far from L_x, then for every*

$$\Pr[D((y \circ \pi)|] < s(|x|) \tag{4}$$

The goal behind proof composition, introduced by Arora and Safra [AS98] is to compose two PCPs to improve the query complexity of the composed PCP system.

First why, do we want to compose PCPs? Recall that our original goal is to construct PCPs with $O(\log n)$ randomness and $O(1)$ query complexity. In last lecture, we constructed PCPs with polynomial randomness and $O(1)$ query complexity. However, any "direct" PCP construction, that we are aware of, that achieves sub-polynomial randomness complexity requires super-constant query complexity. How do we then construct PCPs that satisfy our original goal? Let us look at this problem more closely. As a concrete example, consider the PCPs of Babai et al. [BFLS91] that achieves poly $\log n$ *randomness and* poly $\log n$ *query complexity. Elaborating, the verifier tosses* poly $\log n$ *random coins and produces a set I of* poly*logn indices and a decision circuit D and wishes to check if $D(\pi|_I) = 1$. Is it possible to verify that ""$D(\pi|_I) = 1$" without reading all of $\pi|_I$? On the face of it, this seems impossible. But recall that a PCP does something very similar – it verifies a proof without reading the proof in its entirety. The ingenuous idea of Arora and Safra was to* *use a PCP recursively to check that $D(\pi|_I) = 1$. For the purpose of differentiation, we will call the original PCP, the outer PCP and this new PCP, which is supposed to check that $D(\pi|_I) = 1$ without reading all of $\pi|_I$, the inner PCP. The main idea of proof composition if that the outer verifier V_{out}, instead of checking if $pi|_I$ satisfies the predicate D, transfers control to another "inner" PCP verifier, V_{in} which performs this check with the help of an additional proof.*

The above discussion is very informal. Let us see if we can make this idea work. First notice that the input to the inner verifier V_{in} is the circuit D and the bits $\pi|_I$. By definition, a PCP reads its entire input. Thus, V_{in} read at least $|I|$ bits. This is of no use, as the query complexity has not reduced. This is where PCPs of proximity come to the rescue. Recall that a PCP of proximity verifier gets two inputs, one explicit input and another implicit input, which it has oracle access to. We could thus use a PCP of proximity for the inner verifier V_{in}, which is given the circuit D as explicit input and the proof bits $\pi|_I$ as an implicit input. We could thus compose an outer PCP verifier V_{out} with a inner PCPP verifier V_{in} to reduce the query complexity. Observe that the query complexity of such a composed PCP Verifier is that of the inner verifier, which could be considerably smaller than that of the outer verifier.

Does such a composition make any sense? Syntactically, we can compose a outer PCP Verifier with an inner PCPP verifier. But does the resulting composed PCP Verifier having any meaningful semantics? We note now that merely composing PCP verifier with a PCPP verifier will not necessarily give anything meaningful since the soundness of the outer PCP verifier V_out only guarantees that if the assertion is true, then the proof bits $\pi|_I$ satisfy D and if the assertion is false then $D(\pi|_I) = 0$. However, if the V_{in} is now a "PCP of proximity", we will only be able to distinguish between the inputs π_I that satisfy the Boolean predicate D and those that are "sufficiently" (or δ-)far from a satisfying D. For this purpose, we strengthen the soundness guarantee of the outer PCP verifier as follows: recall the standard soundness guarantee of a PCP verifier.

Soundness: $x \notin L \implies \forall \pi,\ Pr[D(\pi|_I) = 1] < S(|x|)$

We will replace the above soundness requirement by a more stringent one, that we call robust soundness.

Robust) Soundness: $x \notin L, \forall\pi, Pr[\pi|_I$ *is* ρ*-close from* $SAT(D)] < S(|x|)$.

In other words, the local view $\pi|_I$ not only violates the predicate D with probability s, but is in fact ρ-far from satisfying the predicate D with probability s. We call s the soundness error as before and ρ the robustness parameter of the PCP. A PCP that satisfies the robust soundness condition is called a robust PCP (a robust PCPP is defined similarly). If the robustness parameter is ρ, we denote the corresponding PCP as rob-$PCP_{1,1-\epsilon}[r, q, \rho]$. In the case of PCPs of proximity, we refer to the corresponding PCPP as rob-$PCP_{1,1-\epsilon}[r, q, \delta, \rho]$ where δ and ρ are respectively the proximity and robustness parameters respectively.

We are now ready to prove the Composition Theorem, which must be evident from the above discussion. In the following we will refer to the decision complexity of the PCP(P) verifiers, which is basically the size of the decision circuit D output by the corresponding verifier.

Theorem 5.5 (Composition)**.** *Suppose that for functions $r_{out}, r_{in}, d_{out}, d_{in}, q_{in} : N \to N$ and $\epsilon_{out}, \epsilon_{in}, \rho_{out}, \delta_{in} : N \to [0, 1]$,*

- *Language L has a robust PCP verifier V_{out} with randomness r_{out}, decision complexity*

d_{out}, robust-soundness error $1-\epsilon_{out}$ and robustness parameter ρ_{out}

- Circuit-VAL *has PCPP verifier called V_{in} with randomness r_{in} with query complexity q_{in}, decision complexity d_{in}, proximity parameter δ_{in}, and soundness error $1-\epsilon_{in}$*
- $\delta_{in}(d_{out}(n)) \leq \rho_{out}(n), \forall n$

(Composed PCP:) Then, L has a PCP verifier V_{comp} so that

- *randomness complexity $r_{out}(n) + r_{in}(d_{out}(n))$*
- *query complexity $q_{in}(d_{out}(n))$*
- *decision complexity $d_{in}(d_{out}(n))$*
- *soundness error 1 - $\epsilon_{out}(n) \cdot \epsilon_{in}(d_{out}(n))$*

Also, we will show

- *If (instead of being a PCP) the verifier V_{out} is a PCPP with proximity parameter $\delta_{out}(n)$ then the composed verifier V_{comp} is PCPP with proximity parameter $\delta_{out}(n)$*
- *If V_{in} has robust-soundness parameter $\rho_{in}(n)$, then V_{comp} has robust-soundness with robustness parameter $\rho_{in}(d_{out}(n))$.*

Proof. As mentioned in the discussion above, the main idea here is that we use the outer PCP verifier to select the positions in the proof to look at and then transfer control to the inner PCPP which verifies that the positions are close to being accepted by the outer-verifier's decision circuit. Thus, the new proof consists of a proof for the outer verifier as well as new proofs for the inner verifier. Each proof for the inner verifier will correspond to a possible setting of the outer verifier's random coin tosses. We index the positions of the new (combined) oracle by pairs so that (out,i) denotes the i'th position in the part of the oracle that represents the outer verifier's proof oracle, and (R, j) denotes the j'th position in the R-th "auxiliary" block (representing the R-th possible proof, which is associated with the outer verifier's coins). We now formally describe the verifier produced under the composition, which we call $V_{comp}(x)$.

1. Choose $R \leftarrow_R \{0,1\}^{r_{out}}$
2. Run $V_{out}(x;R)$ to obtain $I_{out} = (i_1, ..., i_{q_{out}})$ and D_{out}
3. Run $V_{in}(D_{out})$ (on random coin tosses) to obtain I_{in}=$((b_1, j_1)...(b_{q_{in}}, j_{q_{in}}))$ and D_{in} (Here we are using the convention for a PCP of proximity, that if $b=0$, then the query (b,j) refers to the j-th position in the implicit input and if $b=1$, then the the query (q,j) refers to the j-th position in the proof oracle.
4. For each $l = 1, ..., q_{in}$, determine the queries of the composed verifier
   (a) If b_l=0, set $k_l = (out, i_{j_l})$ i.e., V_{in}'s queries to its input oracle are directed to the corresponding locations in V_{out}'s proof oracle

(b) If $b_l = 1$, set $k_l = (R, j_l)$ i.e., V_{in}'s queries to its R'th possible proof oracle are directed to the corresponding locations in the auxiliary proof.

5. Output $I_{comp} = (k_1, ..., k_{q_{in}})$ and D_{in}.

It is easy to verify that V_{comp} satisfies the above randomness, query complexity, decision complexity, and computational complexities. Now we need to verify completeness and soundness.

Suppose that $x \in L$. Then by completeness of the outer verifier, there exists a proof π_{out} making V_{out} accept with probability 1. In other words, for every $R \in \{0,1\}^{r_{out}}$ if we set $(I_{out}, D_{out}) = V_{out}(x, R)$, we have $D_{out}(\pi_{out}|_{I_{out}}) = 1$. By completeness of the inner verifier, there exists a proof π_R such that $V_{in}(D_{out})$ accepts the oracle $\pi|_{I_{out}} \circ \pi_R$ with probability 1. If we set $\pi_t(t,) = \pi_t()$ for all $t \in \{\text{out}\} \cup \{0,1\}^{r_{out}}$ then V_{comp} accepts π with probability 1.

Suppose that we are in the soundness case, with $x \notin L$. Let π be any oracle. Define oracles $\pi_t()$=$\pi(t,)$. Here is where we use robust soundness. by the robust-soundness property of V_{out} with probability greater than ϵ_{out} over the choices of $R \in \{0,1\}^{r^{out}}$, if we set $(I_{out}, D_{out}) = V_{out}(x; R)$, then $\pi_{out}|_{I_{out}}$ is $\rho_{out}-$far from satisfying from D_{out}. Fixing such an R, by the PCPP-soundness of V_{in}, it holds that $V_{in}(D_{out})$ rejects the oracle $\pi_{out}|_{I_{out}} \circ \pi_R$ with probability greater than ϵ_{in}. Therefore $V_{comp}(x)$ rejects oracle π with probability at least $\epsilon_{out} \cdot \epsilon_{in}$.

We show the next two claims, and they follow easily, If V_{out} is a PCPP verifier, then the input is of the form (x, y) where y is "implicit" or available by oracle access only. In this case we use the same proof above but replace references to the oracle π_{out} with $y \circ \pi_{out}$, and for soundness we should consider the case that y is δ-far from $L(x)$. If V_{in} has robust-soundness, then at the end of the soundness analysis, we note that not only is $\pi_{out}|_{I_{out}} \circ \pi_R$ rejected with greater probability than ϵ_{in} but rather it is ρ_{in}-far from being accepted by V_{in} (and also V_{comp}) □

The notion of proof composition was introduced by Arora and Safra [AS98]. The above modular form of composition is due to Ben-Sasson et al [BGH$^+$06] and Dinur and Reingold [DR06].

5.2.1 Consequences of Composition

Instantiating the inner PCP of proximity with that obtained in Theorem 5.1, we get the following corollary of the Composition Theorem

Corollary 5.6. *There exists δ_0 such that, if $L \in$ rob-$PCP_{1,1-\epsilon}[r, q, \rho]$ then $L \in PCP_{1,1-\epsilon'}[r + O(q^2), 17]$ where $\epsilon' = \epsilon \cdot \min\{\rho, 3\delta_0\}/3$.*

The above corollary shows that any robust PCP (not necessarily constant query) for L can be composed with the exponential sized PCP of proximity for Circuit-VAL *to obtain a constant query PCP for L with at most an $O(q^2)$ increase in randomness. However, to do this we first need to construct a robust PCP for L. We discuss below how this is performed.*

Note that any PCP for L with query complexity q is trivially a robust PCP with robustness parameter $1/q$ as if a local view is rejected, at least 1 bit of the local view (i.e., $1/q$-fraction of the local view) must be changed in order to make it an accepting view. Thus,

$PCP_{1,1-\epsilon}[r,q] \subseteq$ rob-$PCP_{1,1-\epsilon}[r,q,1/q]$. Extending the same idea to a non-Boolean alphabet Σ, we get $PCP^{\Sigma}_{1,1-\epsilon}[r,q] \subseteq$ rob-$PCP^{\Sigma}_{1,1-\epsilon}[r,q,1/q]$. However, we need to convert the PCP into a Boolean alphabet before composing it with the exponential PCPP as in Corollary 5.6. A trivial way to convert a PCP over a non-Boolean alphabet into Boolean string is to encode every symbol of the alphabet Σ using $\log_2|\Sigma|$ bits to obtain $PCP^{\Sigma}_{1,1-\epsilon}[r,q] \subseteq PCP_{1,1-\epsilon}[r,q\cdot a]$ where $a = \log_2|\Sigma|$. However, this trivial conversion ignores the fact that the original non-Boolean PCP has robustness $1/q$. The converted Boolean PCP has robustness only $1/qa$. We can instead perform the following conversion that almost retains the original robustness: Encode every symbol of $|\Sigma|$ using a good error-correcting with constant rate and linear distance. Suppose this encoding converts every symbol of Σ into $C\log_2|\Sigma| = C\log_2 a$ bits and furthermore the distance of the code is μCa. Such a conversion increases the number of bits queried from qa to $O(qa)$ (more precisely qa/μ. However, observe that the robustness of this converted PCP is at least $(\mu Ca)/Cqa = \mu/q$. Hence, we have that the $PCP^{\Sigma}_{1,1-\epsilon}[r,q] \subset$ rob-$PCP_{1,1-\epsilon}[r,O(qa),O(1/q)]$ where $a = \log_2|\Sigma|$. Combining these observations with Corollary 5.6, we have the following proposition.

Proposition 5.7.

$$PCP^{\Sigma}_{1,1-\epsilon}[r,q] \subseteq PCP_{1,1-\epsilon'}[r+O(q^2a^2),17]$$

where $\epsilon' = \epsilon\cdot\min\{O(1/q),\delta_0\}/3$.

Sometimes it is useful to convert the final constant-query PCP (constant =17 in above) which is over the Boolean alphabet into a 2-query PCP (over a larger alphabet). This is typically done by issuing 2 queries — one of which is a query including all queries bundled together and the other one a random query and then checking that the answer to all the queries satisfies the local constraint and that the answer to the random query is consistent with the answers to the bundled query. This converted PCP is over an alphabet $|\Sigma|$ such that $\log_2|\Sigma| = q$. This process however reduces ϵ by a factor of $1/q$. More formally,

$$PCP_{1,1-\epsilon}[r,q] \subseteq PCP^{\Sigma}_{1,1-\epsilon/q}[r+\log_2 q, 2],$$

where Σ is any alphabet such that $\log_2|\Sigma| \geq q$ (see Exercise 1 for more details).

Combining this conversion with Proposition 5.7, we obtain the following:

Lemma 5.8.

$$PCP^{\Sigma}_{1,1-\epsilon}[r,q] \subseteq PCP^{\Sigma'}_{1,1-\epsilon'}[r+O(q^2a^2)+\log_2 17, 2]$$

where $\epsilon' = \epsilon\cdot\min\{O(1/q),\delta_0\}/3\cdot 17$, $a = \log_2|\Sigma|$ and Σ' is any alphabet such that $\log_2|\Sigma'| \geq 17$.

The above lemma will be used in the final phase (alphabet reduction) of Dinur's proof of the PCP Theorem.

References

[AS98] Sanjeev Arora *and* Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM, 45(1):70–122, January 1998. (Preliminary Version in* 33rd FOCS*, 1992).* *doi:10.1145/273865.273901.*

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, *and* Mario Szegedy. Checking computations in polylogarithmic time. *In* Proc. 23rd ACM Symp. on Theory of Computing (STOC), *pages 21–31. New Orleans, Louisiana, 6–8 May 1991.* *doi:10.1145/103418.103428.*

[BGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, *and* Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM J. Computing, 36(4):889–974, 2006. (Preliminary Version in* 36th STOC, *2004).* *doi:10.1137/S0097539705446810.*

[DR06] Irit Dinur *and* Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP Theorem. *SIAM J. Computing, 36:975–1024, 2006. (Preliminary Version in* 45th FOCS, *2004).* *doi:10.1137/S0097539705446962.*