Advanced Proof-Systems - Problem Set 3

Yosef Goren

December 31, 2022

1 MIP vs PCP

1.1

Denote the concatenation of m_1 and m_2 with $m_1||m_2$.
Let $L \subseteq \{0,1\}^*$ be a language with a $k-prover$, $2-message\ MIP$.
Since this protocol is a $2-message\ MIP$, we know the structure of communications: in the first round the verifier sends a message to all provers, and in the second round each prover sends a message to the verifier.

Let $x \in L$, in the run of the protocol on x, Denote the message sent by the V to P_i with $m_{q,i}$, denote the message sent by P_i to V with $m_{r,i}$.

The set of possible values of $m_{q,i}$ is bounded with it's maximal size. Each such value will yield an appropriate $m_{r,i}$ response, which is independet of any other $m_{q,j}$ 'queries' sent by the verifier.
Thus for each $i \in [n]$ we can define a function from query values to response values; for any query value we can define the response value to be 0. Thus we are left with a function $Resp_i : \{0,1\}^{l_V} \to \{0,1\}^{l_P}$, where $\forall m_{q,i} \forall, Resp_i(m_{q,i}) = m_{r,i}$.

For the purposes of accessing the correct response later we can also pad the response values with 0's so that all of them are exactly l_P in length.

Define the PCP proof of x as:

$$PCP_x := Resp_0(0)||Resp_0(1)||\dots||Resp_0(2^{l_V}-1)$$
$$||Resp_1(0)||Resp_1(1)||\dots||Resp_1(2^{l_V}-1)$$
$$...$$
$$||Resp_{k-1}(0)||Resp_{k-1}(1)||\dots||Resp_{k-1}(2^{l_V}-1)$$

Note $|Resp_i(j)| = l_P$, thus each line in the definition of PCP_x is equal to $l_p \cdot 2^{l_V}$. And the whole size is equal to $k \cdot l_p \cdot 2^{l_V}$.

The PCP verifier V_p with be based on the MIP verifier V_m.
On the run of $V_p(x, PCP_x)$, V_p will first use V_m to ask it what queries to make.
For each query $m_{q,i}$ to P_i made by V_m, V_p will look at:

$$PCP_x[l_P \cdot (2^{l_V} \cdot i + m_{q,i}) : l_P \cdot (2^{l_V} \cdot i + m_{q,i}) + l_P]$$

Or in other words, the bits corresponsing to the response for $m_{q,i}$ in P_i's section of PCP_x.

After the PCP verifier gets all these bits, it gives them back to V_m as the responses to the queries - and accepts iff V_m accepts.

- Completness:
  If $x \in L$ - our PCP_x is well defined. And for whatever query V_m makes - it recives the exact response it should get from P_i - thus since it will accept on P_i's responses - it will accepts on the messages sent by V_p, meaning V_p will accept on x.

- Soundness:
  Let there be a set of PCP_x values (defined also on $x \notin L$). WLOG $\forall x, |PCP_x| = l_P$ - since it is easy for V to check if that is the case.
  Thus we can esaly use this set of PCP_x's to construct a set of 'mallicious' provers $P_0, ..., P_{k-1}$: Each P_i on query $m_{q,i}$ will response with:

  $$PCP_x[l_P \cdot (2^{l_V} \cdot j + m_{q,i}) : l_P \cdot (2^{l_V} \cdot j + m_{q,i}) + l_P]$$

  Now for each instance where $V_p(x, PCP_x) = 1$, we have $(V_m, P_0, ..., P_{k-1})(x) = 1$ since the only way for V_p to accept is if V_m does - and it runs on the same inputs (including randomizations) in both cases.
  Thus:

  $$\Pr[V_p(x, PCP_x) = 1] \leq \Pr[(V_m, P_0, ..., P_{k-1})(x) = 1] \leq \frac{1}{2}$$

1.2

Let $L \subseteq \{0,1\}^*$ be a language with a PCP verifier V_p set of proofs bounded by in length m.
Define the following $q - prover\ MIP$ for it:
The protocol on input x:

- Verifier V_m:
  1. sample $B \leftarrow^{\$} \{0,1\}$.
  2. if $B = 0$ (verify PCP):
     (a) Get the set of queries $Q = \{Q_i \mid i \in [q]\}$ from V_p on input x.
     (b) For all $i \in [q]$, send Q_i to P_i.
     (c) Denote the bit returned by P_i with b_i.
     (d) Verify V_p's acceptence on query results $\{b_i \mid i \in [q]\}$.
  3. if $B = 1$ (Verify consistency):
     (a) Sample $r \leftarrow^{\$} [m]$.
     (b) For all $i \in [q]$, send r to P_i.

(c) Denote the bit returned by P_i with b_i.
(d) Verify $b_i = b_j, \forall i, j \in [q]$.

- Prover P_i:

1. Recive an index i from the verifier.
2. Return $PCP_x[i]$.

Correctness:

- Complexity:
The integer representation of each query is of size $log(m)$, thus it is the length of the messages sent by V_m.

- Completness:
Let $x \in L$. Denote with $PCP_x[Q]$ the set of bits corresponding to the queries Q in PCP_x.
In the standard usage of the PCP verifier V_p - it will recive $PCP_x[Q]$ as the responses to the queries Q - and since it has perfect completness (WLOG as seen previously in the course), it will accept.
When V_m invokes V_p - it sends it the same $PCP_x[Q]$, thus V_p accepts here too - and so does V_m.

- Soundness:
Let $x \notin L$.
Let $\{P_i^* \mid i \in [q]\}$ be a set of (possibly mallicious) provers.

Now we use these provers to construct PCP_x:
PCP_x has a section corresponding to each prover P_i^*, and in each section - the j'th bit corresponds to the response of P_i^* on query j which is the most probable. If there is more than one most probable response select the minimal one.

As a shorthand denote $P_1^*, ..., P_q^*$ as PS^*.

Denote with $C_{i,j}$ the event that P_i returns the most probable response on query j and $c_{i,j} = \Pr[C_{i,j}]$. Since there are only two possible responses (1 and 0) - $c_{i,j} \geq \frac{1}{2}$.

In the event that $B = 0$:
Let $C := \bigwedge_{i \in [q]} C_{i,Q_i}$ - meaning the event that all provers were consistent with PCP_x. Let $c = \Pr[C]$.
Since we know the provers cannot communicate - the probabilities for their selections are independent. Hence $c = \mathbb{E}_Q[\prod_{i \in [q]} c_{i,Q_i}]$.

$$\Pr[(V_m, PS^*)(x) = 1 \mid B = 0]$$

$$= \Pr[(V_m, PS^*)(x) = 1 \mid B = 0 \wedge C] \Pr[C] + \Pr[(V_m, PS^*)(x) = 1 \mid B = 0 \wedge \neg C] \Pr[\neg C]$$
$$= c \Pr[(V_m, PS^*)(x) = 1 \mid B = 0 \wedge C] + (1-c) \Pr[(V_m, PS^*)(x) = 1 \mid B = 0 \wedge \neg C]$$
$$= c \Pr[V_p(x, PCP_x) = 1] + (1-c) \Pr[(V_m, PS^*)(x) = 1 \mid B = 0 \wedge \neg C]$$
$$\leq c \Pr[V_p(x, PCP_x) = 1] + (1-c) \cdot 1 = 1 + c(\Pr[V_p(x, PCP_x) = 1] - 1)$$

In the event that $B = 1$:
Let $C := \bigwedge_{i \in [q]} C_{i,r}$ and $c' := \Pr[C]$.
Thus $c = \frac{1}{q} \sum_{r \in [q]} \prod_{i \in [q]} c_{i,r}$.

$$\Pr[(V_m, PS^*)(x) = 1 \mid B = 1]$$
$$\Pr[(V_m, PS^*)(x) = 1 \mid B = 1 \wedge C] \Pr[C] + \Pr[(V_m, PS^*)(x) = 1 \mid B = 1 \wedge \neg C] \Pr[\neg C]$$
$$= c \Pr[(V_m, PS^*)(x) = 1 \mid B = 1 \wedge C] + (1-c) \Pr[(V_m, PS^*)(x) = 1 \mid B = 1 \wedge \neg C]$$
$$= c \cdot 1 + (1-c) \cdot 0 = c$$

Thus:

$$\Pr[(V_m, PS^*)(x) = 1] = \frac{1}{2} \Pr[(V_m, PS^*)(x) = 1 \mid B = 0] + \frac{1}{2} \Pr[(V_m, PS^*)(x) = 1 \mid B = 1]$$
$$\leq \frac{1 + c(\Pr[V_p(x, PCP_x) = 1] - 1)}{2} + \frac{c}{2} = \frac{1 + \Pr[V_p(x, PCP_x) = 1]}{2}$$

Hence since V_p has constant soundness - V_m has constant soundness too.

2 Tensor Codes

2.1 A Characterization

$\underline{C_1 \otimes C_2 \subseteq \mathbb{M}}$:
Let $\mathbb{M} \subseteq \mathbb{F}^{n_1 \times n_2}$ be the set of matrices such that all rows are codewords of C_2 and all columns are codewords of C_1.

As defined, we know that $C_i : \mathbb{F}^{k_i} \longrightarrow \mathbb{F}^{n_i}$. Furthermore, we know C_i is a linear function. This means that C_i can be represented as a matrix, or in other words, we can describe $C_i(v) = C_i \cdot v, \forall v \mathbb{F}^{k_1}$.
Furthermore, if we have a matrix $A \in \mathbb{F}^{n_1 \times X}$, applying C_2 to all of it's columns is equivalent to multiplying A by C_2 as in $C_2 \cdot A$. If we want to do the equivalent operation for the rows, we can transpose A, then transpose the result after multiplication, meaning that for any matrix $A \in \mathbb{F}^{x \times n_2}$, applying $(C_1 \cdot (A^T))^T$ is equivalent to applying C_1 to all of A's rows.

Finally, if we want to first apply C_2 to all columns, then apply C_1 to all rows of a matrix $A \in \mathbb{F}^{n_1 \times n_2}$ - we can describe this operation as: $C_1(C_2 A^T)^T$, meaning (*):

$$(C_1 \otimes C_2)(A) = C_1(C_2 A^T)^T = C_1(A C_2^T) = (C_1 A) C_2^T = (C_2 (C_1 A)^T)^T$$

Let $B \in C_1 \otimes C_2$. By def. $\exists A : (C_1 \otimes C_2)(A) = B$. From (*), this means $B = C_1(AC_2^T)$, or in other words $\exists D = AC_2^T$, s.t. $B = C_1 D$ and so each column of B is the result of the C_1 on a column of D - meaning it is a leagal codeword of C_1.
Similarly, from (*) we know $\exists D = C_1 A$ s.t. $B = (C_2 D^T)^T$ - meaning every row is a legal codeword of C_2.
Hence $B \in \mathbb{M}$. In other words; $C_1 \otimes C_2 \subseteq \mathbb{M}$.

$\underline{\mathbb{M} \subseteq C_1 \otimes C_2}$:

$\underline{C_1 \otimes C_2 = \mathbb{M}}$:
An anti-symmetric relation $R \subseteq A \times B$ is a relation that obays:

$$\forall a \in A, \forall b \in B : (a, b) \in R \wedge (b, a) \in R \Rightarrow a = b$$

Lemma: '$\subseteq$' is anti-symmetric:
Let A, B be two sets s.t. $A \subseteq B \wedge B \subseteq$. By def. this means:

$$\forall a \in A, a \in B \wedge \forall b \in B, b \in A$$

Thus $\forall x : x \in A \iff x \in B$ and so $A = B$.
Hence, $\subseteq$ is anti-symmetric.

Thus, since we have shown that $C_1 \otimes C_2 \subseteq \mathbb{M}$ and also $\mathbb{M} \subseteq C_1 \otimes C_2$, we can conclude that $C_1 \otimes C_2 = \mathbb{M}$.

But most importantly, we have shown we remember Matka!

2.2 Rate

We know the input size of $C_1 \otimes C_2$ is $k_1 \cdot k_2$ and the outpus size is $n_1 \cdot n_2$. Thus the rate is: $r = \frac{k_1 \cdot k_2}{n_1 \cdot n_2} = \frac{k_1}{n_1} \cdot \frac{k_2}{n_2} = r_1 \cdot r_2$

2.3 Distance

Let $A \in \mathbb{M}$.
By def. $hw(A) \geq \delta_1 n_2 n_1$, $hw(A) \geq \delta_2 n_1 n_2$. Thus:

$$hw(A) \geq max(\delta_1 n_2 n_1, \delta_2 n_2 n_1) \Rightarrow hw(A) \geq max(\delta_1, \delta_2) \cdot n_1 n_2 \Rightarrow \frac{hw(A)}{n_1 n_2} \geq max(\delta_1, \delta_2) \geq \delta_1 \delta_2$$

Since $\delta = \frac{hw(A)}{n_1 n_2}$ we have $\delta \geq \delta_1 \delta_2$.

2.4 Tensor of Reed-Solomon

2.5 Sumcheck for Tensor Codes

Given an input c, α:
Define $w \in \mathbb{F}^n$ s.t. for all $\lambda \in S$,

$$w[\lambda] := \sum_{i_2, i_3, \ldots, i_d \in S} c(\lambda, i_2, ..., i_d)$$

Let $v := C_{base}^{-1}(w)$.

The protocol:

- Verifier:
  1. Recive v as v'.
  2. Compute $w' := C_{base}(v')$.
  3. Verify $\sum_{\lambda \in S} w'[\lambda] = \alpha$.
  4. If $d = 1$:
     (a) Verify $w' = c$ (by reading c).
  5. Otherwise:
     (a) Sample $\lambda \xleftarrow{\$} [n]$.
     (b) Verify $w'[\lambda] = w[\lambda]$, recursively with α being $w'[\lambda]$, c being $c(\lambda, :, ..., :)$ (the left c is the parameter of the recursively invoked protocol and the right c is the paraneter of the invoking protocol) and d being $d - 1$.

- Prover:
  1. Send v.
  2. Recursively follow the verifier's protocol.

Correctness:

- Complexity:
  The base cases's complexity is $O(log|\mathbb{F}| \cdot n)$.
  Each recusive step also requires $O(n)$ opeartions, where each one is an operation in the field $\mathbb{F}$ and thus can be done with complexity $log|\mathbb{F}|$, so the total complexity of a step is $O(n \cdot log|\mathbb{F}|)$ and the total complexity is $O(n \cdot d \cdot log|\mathbb{F}|)$.

- Completness:
  The base case is immidiate. And recursively:
  If c, α satisfy the condition and P sends $v' = v$.
  So $w' = w$. As we know w satisfies the formula and so does w' and the

verifier passes step 2., with the inductive assumption for completness - the verifier will also accept for $w'[\lambda]$ for any $\lambda \in [n]$ and the verifier passes 4.b.

- Soundness:
  In the case $d = 1$ the checking is exhaustive, so the protocol is perfectly sound, so the soundness error is $1 \geq 1 - \delta$.

  Let us assume a soundness error $1 - \delta^{d-1}$ for the protocol of order $d - 1$. Now consider an instance of protocol with order d on input c, α that do not satisfy the condition.

  If $v' = v$, the verifier will disqualify the proof w.p. 1 at step 2 (*).

  If $v' \neq v$:

  1. Since $v' \neq v$, $w' \neq w$.
  2. Since δ is the minimal relative distance of C_{base}:
  $$\forall a, b \in C_{base} : a \neq b, \Pr_{\lambda \leftarrow [n]}[a(\lambda) \neq b(\lambda)] \geq \delta$$
  3. Due to 1. and 2.:
  $$\Pr_{\lambda \leftarrow [n]}[w'(\lambda) \neq w(\lambda)] \geq \delta$$
  4. If λ is sampled s.t. it satisfies the expression at 3., the verifier will thus disqualify the proof w.p. at-least $1 - \delta^{d-1}$ at step 4.b. due to the inductive assumption.
  5. Thus the chance that the verifier will accept is no more than $\delta \cdot \delta^{d-1} = \delta^d$ (**).

  Now consider the total probability of the verifier accepting the proof:
  $$\Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1]$$
  $$= x \cdot \Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1 \mid v' = v] + (1 - x) \cdot \Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1 \mid v' \neq v]$$
  $$\leq \Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1 \mid v' = v] + \Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1 \mid v' \neq v]$$
  $$=_{(*)} 0 + \Pr_{\lambda \leftarrow [n]}[(V, P)(c, \alpha) = 1 \mid v' \neq v]$$
  $$\leq_{(**)} 1 - \delta^d$$
  Hence we have a soundness error of $1 - \delta^d$.