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:

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

2 Tensor Codes

Yosef Goren

December 26, 2022