Complexité avancée
TD 12

Cristina Sirangelo - LSV, ENS-Cachan

December 17th, 2014

Definition (Multi-prover interactive protocols). Let $P_1, \ldots, P_k$ be infinitely powerful machines whose output is polynomially bounded. Let V be a probabilistic polynomial-time machine. V is called the *verifier*, and $P_1, \ldots, P_k$ are called the *provers*.

A round of a multi-prover interactive protocol on input x consists of an exchange of messages (i.e. words over a given alphabet) between the verifier and the provers, and works as follows :

- The verifier V is executed on an input consisting of x, the history of all previous messages exchanged with all provers (both sent and received messages), and a random tape content of size polynomial in $|x|$. The output of the verifier is computed in time polynomial in $|x|$, and consists of messages to some or all of the provers.
- Each message q_i sent from the verifier to prover P_i is followed by an answer a_i, of size polynomial in $|x|$, sent from the prover P_i to the verifier. The answer a_i is computed by P_i on input consisting of x and the history of all messages previously exchanged between the verifier and the prover P_i (and only P_i).
- Alternatively the verifier may decide not to produce messages, and terminates the protocol by either accepting or rejecting, based on the input x and the history of all previous messages exchanged with all provers.

You can view the protocol as executed by the verifier sharing communication tapes with each P_i, where different provers P_i and P_j have no tapes they can both access, besides the input tape. In a round the verifier stores each message q_i to prover P_i on the i-th communication tape, shared between the prover and P_i. The answer of P_i is put on tape i as well. The verifier has access to the input and all communication tapes, while each prover P_i has access only to the input and tape i.

$P_1, \ldots, P_k$ and V form a *multi-prover interactive protocol* for a language L if the execution of the protocol between V and $P_1, \ldots P_k$ terminates after a polynomial number of rounds (in the size of the input x) and :

- if $x \in L$, then $Pr[(V, P_1, \ldots, P_k) \text{ accepts } x] > 1 - 2^{-q(n)}$;
- if $x \notin L$, then for all provers $P'_1, \ldots, P'_k$, $\quad Pr[(V, P'_1, \ldots, P'_k) \text{ accepts } x] < 2^{-q(n)}$;

where q is a polynomial and the probability is computed over all possible random choices of V.

In this case, we denote $L \in \mathbf{MIP}_k$. The number of provers k need not be fixed and may be a polynomial in the size of the input x. We say that $L \in \mathbf{MIP}$ if $L \in \mathbf{MIP}_{p(n)}$ for some polynomial p. Clearly $\mathbf{MIP}_1 = \mathbf{IP}$, but allowing more provers makes the interactive protocol model potentially more powerful.

Exercice 1. Characterization of MIP. Prove the following characterizations of the class **MIP**.

1. Let M be a probabilistic polynomial-time Turing machine with access to a function oracle. A language L is accepted by M iff :
   - if $x \in L$, then there exists an oracle O s.t. M^O accepts x with probability greater than $1 - 2^{-q(n)}$;
   - if $x \notin L$, then for any oracle O', $M^{O'}$ accepts x with probability smaller than $2^{-q(n)}$.

   Show that $L \in MIP$ if and only if L is accepted by a probabilistic polynomial time oracle machine.
2. Show that $MIP = MIP_2$.

Exercise 2. PCP, MIP and NEXPTIME Prove that

$$\bigcup_{R(n),Q(n),T(n) \text{ polynomials}} \mathbf{PCP}(R(n),Q(n),T(n)) \subseteq \mathbf{MIP} \subseteq \mathbf{NEXPTIME}$$

(**Hint.** It is possible to prove (but you are not required to) that, as with **IP**, one can equivalently use *perfect completeness* in the definition of **MIP**. That is, in the case $x \in L$, we require that the protocol accepts with probability 1, rather than at least $1 - 2^{-q(n)}$. In this exercise use the definition of **MIP** with perfect completeness, and the corresponding notion of probabilistic oracle machine.)

Remark. Indeed **MIP** and this version of **PCP** *coincide* with **NEXPTIME**, but you are not required to prove the opposite inclusions.