Lesson 10: Interactive proofs

Theme: The class **IP**, **MA** and **AM**.

1 The class IP

Let $\Sigma = \{0, 1\}$ and # be a symbol. Let $P : (\Sigma \cup \{\#\}) \to \Sigma^*$ be an *arbitrary* function. Let V be a probabilistic TM whose inputs are of the form:

$$w \ \# \ u_1 \ \# \ v_1 \ \# \ u_2 \ \# \ v_2 \ \# \ \cdots \ \# \ u_m \ \# \ v_m$$

where $w, u_1, v_1, \ldots, u_m, v_m$ are all strings from Σ^*. The outcome of V can be *accept*, *reject* or "send a string u to P."

The function P is usually called the *prover* and V the *verifier*. The interaction between P and V, denoted by (P, V), on input word $w \in \Sigma^*$ consists of rounds defined as follows.

(Round 1:)

- Run V on w.

  If it accepts/rejects, then the interaction stops.

  If the outcome is "sends a string u_1 to P," then V sends u_1 to P.

- Let $P(w\#u_1) = v_1$.

  Then, P sends v_1 to V, and the interaction continues to round 2.

(Round 2:)

- Run V on $w\#u_1\#v_1$.

  If it accepts/rejects, then the interaction stops.

  If the outcome is "sends a string u_2 to P," then it sends u_2 to P.

- Let $P(w\#u_1\#v_1\#u_2) = v_2$.

  Then, P sends v_2 to V, and the interaction continues to round 3.

and so on. The interaction continues until V accepts/rejects, in which case we say that the interaction (P, V) accepts/rejects w.

On each round i, the verifier V starts with its initial state and the position of its head is on the first position of $w\#u_1\#v_1\#\cdots\#u_{i-1}\#v_{i-1}$. On each round i, we call the string u_i *the verifier's query* and v_i *the prover's reply.*

Remark 10.1 We usually assume that V runs in polynomial time in *the length of the input word* w. That is, there is a polynomial $p(n)$ such that on each round i the run time of V on $w\#u_1\#v_1\#\cdots\#u_{i-1}\#v_{i-1}$ is bounded by $p(|w|)$.

In this case we may assume that V always tosses the random string r before round 1 starts and in each round i, the verifier V is a deterministic TM with input $(w, r)\#u_1\#v_1\#\cdots\#u_{i-1}\#v_{i-1}$. Moreover, the length of each reply v_i is also bounded by the $p(|w|)$ and so is the number of rounds in the interaction.

Note also that the prover P does not know the random string r. He only knows the input word and the queries sent by the verifier.

Definition 10.2 A (polynomial time) verifier V decides a language L, if for every word $w \in \Sigma^*$, the following holds.

- If $w \in L$, then there is a prover P such that $\mathbf{Pr}_r[\ (P,V) \text{ accepts } w\ ] \geqslant 2/3$.
- If $w \notin L$, then for every prover P, $\mathbf{Pr}_r[\ (P,V) \text{ accepts } w\ ] \leqslant 1/3$.

The class **IP** is defined as $\mathbf{IP} \stackrel{\mathsf{def}}{=} \{L \mid \text{there is a polynomial time verifier } V \text{ that decides } L\}$.

Example 10.3 We will consider the interactive proofs for following two languages.

- $\mathsf{NON\text{-}ISO} \stackrel{\mathsf{def}}{=} \{(G_0, G_1) \mid G_0 \text{ is not isomorphic to } G_1\}$.
- $\mathsf{NON\text{-}SQ} \stackrel{\mathsf{def}}{=} \{(a,p) \mid a \not\equiv b^2 \pmod p \text{ for some } b \text{ where } p \text{ is a prime}\}$.

Lemma 10.4 IP $\subseteq$ PSPACE.

2 The class MA and AM

The class AM. The *Arthur-Merlin* (**AM**) class is defined as the class **IP** with additional restrictions. On input w, it does the following.

- V generates a random string r and sends it to P.
- P replies with a string p.
- V runs a deterministic computation on input w, r, p.

  That is, V is not allowed to use any random string except r.

The class MA. The *Merlin-Arthur* (**MA**) class is defined as the class **IP** with additional restrictions. On input w, it does the following.

- P sends a string p to V.
- Run V on input w, p, where V is a polynomial time PTM.

  Here V is allowed to generate some random string.

Note that in the class **AM** and **MA** the interaction consists of only one round. It can be easily generalized to multiple rounds.

Theorem 10.5

- $\mathbf{AM} \subseteq \mathbf{\Sigma}_3^p$.
- $\mathbf{MA} \subseteq \mathbf{\Sigma}_2^p$.

Theorem 10.5 can be proved using the same technique as Theorem 7.4.