Notes for Lecture 14

In this lecture we will look at the notion of *Interactive Proofs* and the induced class **IP**. Intuitively, this class is obtained by adding the ingredients Interaction and Randomness to the class **NP**. As we shall see, adding either one does not make much difference but the two put together make **IP**a very powerful class.

1 Interactive Proofs

We begin be recalling the definition of **NP**. Given a language L, we say that: $L \in \mathbf{NP}$ iff there is an algorithm $V(\cdot, \cdot)$ running in polynomial time $T(\cdot)$ and a polynomial $p(\cdot)$ such that

$$x \in L \Leftrightarrow \exists y, |y| \leq p(|x|) \text{ and } V(x, y) \text{ accepts}$$

We can rewrite the above as two separate conditions, using elementary ideas from logic.

$$x \in L \Rightarrow \exists y, |y| \leq p(|x|) \text{ and } V(x, y) \text{ accepts} \qquad \text{(Completeness)}$$
$$x \notin L \Rightarrow \forall y, |y| \leq p(|x|) \text{ and } V(x, y) \text{ rejects} \qquad \text{(Soundness)}$$

We may generalise this to the figure below 1 where the process of recognizing if $x \in L$ is carried out by a *Verifier* V that must run in polynomial time, and a *Prover* P, who has unbounded time and space. The task of the prover is to convince the verifier that $x \in L$ and the verifier is a skeptic who's task is to be convinced if and only if the string actually belongs in the language.

Now the class **NP** is the set of languages L such that $L \in \mathbf{NP}$ iff there is a Prover P and a polynomial time verifier V such that:

$$x \in L \Rightarrow P \text{ has a strategy to convince V} \qquad \text{(Completeness)}$$
$$x \notin L \Rightarrow P \text{ has no strategy to convince } V \qquad \text{(Soundness)}$$

By *strategy* we mean in this case the certificate or proof that is polynomially small, (corresponding to y in the figure 1) that P supplies to V, while later we will generalise it to interaction *i.e.* where a sequence of messages is exchanged between the two and a strategy for P is a function from the sequence of messages seen to the next message that P must send to V.

Thus, we generalise the above definition to get the class **IP**. The class **IP**differs in the following two ways:

1. There is randomness, the verifier may be a randomized machine.
2. There is interaction: unlike the above where a single static proof is sent across, there are rounds of communication, where the verifier may "ask" further questions of the prover based on the messages that have been sent to it.

First, we shall see that both of the above are required.

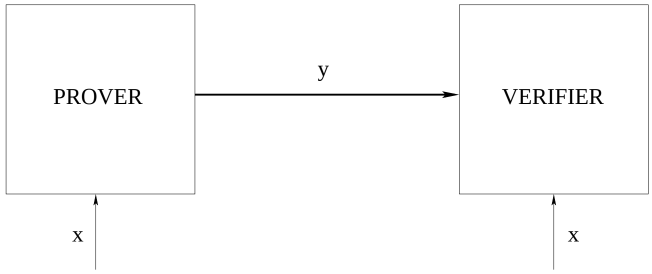

Figure 1: **NP**with a Prover-Verifier mechanism

1.1 NP+ Interaction = NP

Let the class $\overline{\mathbf{NP}}$ be the class corresponding to $\mathbf{NP}$ + interaction. That is we allow the verifier to ask questions and thus the strategy is a function from sequences of messages (the past communication) to messages (the next message that the verifier must send). We say that the class $\overline{\mathbf{NP}}$ is the set of languages L such that $L \in \overline{\mathbf{NP}}$ iff there is a Prover P and a polynomial time verifier V such that:

$$x \in L \Rightarrow P \text{ has a strategy to convince } V$$
$$x \notin L \Rightarrow P \text{ has no strategy to convince } V$$

It is easy to check that $\mathbf{NP} = \overline{\mathbf{NP}}$. We only need to show $\overline{\mathbf{NP}} \subseteq \mathbf{NP}$, as the other inclusion is obvious. Say $L \in \overline{\mathbf{NP}}$, let P, V be the prover and verifier for L. Define P', V' the one-round prover and verifier as follows: P' sends to V' the entire transcript of the interaction between P, V as the certificate. Suppose that interaction was the set of messages $y_1, \ldots, y_n$ as in the figure 2 below. V' is essentially like V only it has been given all the answers to its questions at once. Thus it takes the first message of P, y_1, its response (behaving as V) is exactly y_2, but y_3 is the reply to that, and so on, it "asks" the questions and is supplied all the answers on the same transcript, and it accepts iff after this exchange V accepted. Thus the language accepted by P', V' is the same as L and so $L \in \mathbf{NP}$. Note that the entire set of messages is polynomially long as there are polynomially many messages and each is polynomially long.

Note that for the above to work, the prover must at the very beginning know what responses V may make, which is possible only if V is deterministic. If V was randomized then to cover all possible questions in a polynomially long interaction P' would have to send an exponentially large transcript.

1.2 NP + Randomness

By adding randomness to the class **NP** we get the class **MA**.

Definition 1 (MA) $L \in \mathbf{MA}$ *iff there exists a probabilistic polynomial time machine* V

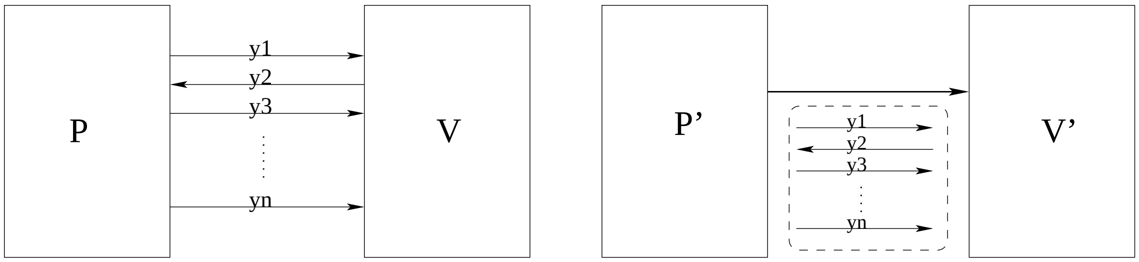

Figure 2: Simulating several rounds of interaction in one round

such that:

$$x \in L \Rightarrow \exists y.\mathbf{Pr}[V(x,y) \ \textit{accepts}\ ] \geq \frac{2}{3}$$
$$x \in L \Rightarrow \forall y.\mathbf{Pr}[V(x,y) \ \textit{accepts}\ ] \leq \frac{1}{3}$$

As in similar classes, the success probabilities may be boosted arbitrarily high. It is conjectured that **MA** = **NP**. it is known that if **coNP** $\subseteq$ **MA** then the polynomial heirarchy collapses.

We are now in a position to define formally the class **IP**.

2 IP

Definition 2 (IP) *A language L is in* $\mathbf{IP}(r(\cdot))$ *iff there exists a probabilistic polynomial time verifier V such that:*

$$x \in L \Rightarrow \exists P.\mathbf{Pr}[V \ \textit{interacting with}\ P \ \textit{accepts}\ ] \geq \frac{2}{3}$$
$$x \in L \Rightarrow \forall P.\mathbf{Pr}[V \ \textit{interacting with}\ P \ \textit{accepts}\ ] \leq \frac{1}{3}$$

Where V also uses at most $r(|x|)$ rounds of interaction.

Note that by *round* we really mean a single “message” and not a question-answer sequence. That is, by two rounds we mean that the Verifier asks a question of the Prover and the Prover replies, which is exactly two messages.

A related class is **AM** which is defined as follows:

Definition 3 (AM) *A language L is in* $\mathbf{AM}(r(\cdot))$ *iff* $L \in \mathbf{IP}(r(\cdot))$ *and at each round the verifier sends a random message, that is a message that is completely random and independent of all previous communication.*

The above is also known as *Public Coin Proof Systems*.

There are the following surprising theorems about the classes **AM** and **IP**.

Theorem 1 $\mathbf{IP}(r(n)) \subseteq \mathbf{AM}(r(n)+2)$

Theorem 2 $\forall r(n) \geq 2$, $\mathbf{AM}(2r(n)) \subseteq \mathbf{AM}(r(n))$.

This yields the following corollaries:

Corollary 3 $\mathbf{AM}(O(1)) \subseteq \mathbf{AM}(2)$

That is, all constant round **AM** proofs can be done in just two rounds. And also:

Corollary 4 **IP**$(O(1))$=**AM**$(O(1))$=**AM**(2)

Finally we have the famous theorem of Shamir:

Theorem 5 $\mathbf{IP}(poly(n)) = \mathbf{PSPACE}$

We know that **AM**(2) is good enough for systems with a constant number of rounds, but it would be surprising indeed if polynomially many rounds could be simulated in a constant number of round. Due to the above result, when people say **AM**they mean **AM**(2), but when people say **IP** they mean **IP**(poly).

The above classes **IP**and **AM** were discovered independently. The first was discovered by Shafi Goldwasser, Silvio Micali and Charles Rackoff. They proposed the class **IP** in order to facilitate the definition of the class Zero Knowledge Proofs which we shall see shortly, the paper was written in 1983 but appeared first in FOCS 85. The class **AM** was discovered by Lazlo Babai (the journal version of the paper is by Babai and Moran), in order to characterize the complexity of a group-theoretic problem. The paper also appeared in FOCS 85, and contained theorem 2.

There is also the following theorem that gives an upper bound on **IP** with constant rounds.

Theorem 6 *If* $\mathbf{coNP} \subseteq \mathbf{IP}(O(1))$ *then the polynomial heirarchy collapses.*

Thus for a long time it was believed that **IP**could not capture **coNP**as it was thought that **IP**with a constant number of rounds was roughly as good as **IP**with polynomially many rounds. However, it was shown that in fact **coNP**was contained in **IP**(poly(n)), the proof of which we shall shortly see and a few weeks after that result, Shamir proved that in fact **IP** was the same as **PSPACE**.

3 An Example: Graph Non-Isomorphism

To get a feel of the power of interaction and randomness let us look at an example, Graph Non-Isomorphism. The problem is the following:

- Input: Graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
- Output: Decide whether there exists *NO* $\pi : V_1 \mapsto V_2$ such that $(u, v) \in E_1$ iff $(\pi(u), \pi(v)) \in E_2$.

This problem is not known to be in **NP**. The difficulty lies in constructing a succint proof that no permutation of the vertices is good, given that there are exponentially many permutations, let us see how this is in **AM**(2).

Suppose that the skeptic Arthur (the verifier) has the two graphs G_1, G_2 and wishes to know whether or not all the permutations are bad. His only resources are randomness and his wizard Merlin who is all knowing (the prover).

He does the following:

1. Picks a random $b \in \{1, 2\}$.
2. Creates G to be a random permutation of G_b.
3. Sends G to Merlin and demands to know of Merlin which graph he sent him a permutation of.
4. If Merlin was correct Arthur believes that the graphs are non-isomorphic (accepts), otherwise Merlin was incorrect, Arthur believes the graphs are isomorphic (rejects).

Notice that if the graphs are really non-isomorphic, then Merlin will always be able to tell which graph was sent to him as only one of the graphs will be isomorphic to the random permutation that Merlin is given, and with his infinite resources Merlin will figure out which one and thus will reply correctly and so Arthur will accept with probability 1.

If the graphs are isomorphic then Merlin cannot tell as the permutation is a permutation of both G_1, G_2. Merlin's output depends only on G_1, G_2, G, say it is i. Now $i = b$ with probability $\frac{1}{2}$ as b is picked independantly at the start. Thus in the case the graphs are isomorphic, Arthur accepts only with probability $\frac{1}{2}$. By repeating this k times we can shrink this to 2^{-k}.

Thus we have a protocol, a prover and a verifier for Graph Non-Isomorphism and so it is in **AM**(2).

Note: The verifier of the above has learnt nothing at all about the Graphs that he did not know before. He does not have a proof or certificate with which he may convince a third party that the Graphs are non-isomorphic. The entire power was in the interaction between it and the prover, and the one bit that the prover gave it gives it significant confidence but no *knowledge*. This was therefore a *Zero Knowledge* Proof.

Remark 1 Graph Non-Isomorphism $\in$ **IP**(2) = **AM**(2). *Thus,* Graph Isomorphism *is not* **NP***-complete unless the polynomial heirarchy collapses. Thus is because, if* Graph Isomorphism *is* **NP***-complete then* Graph Non-Isomorphism *is* **coNP***-complete, and from the above result,* **coNP** $\subseteq$ **AM**(2) *and from the theorem 6, the polynomial heirarchy would collapse.*

4 Random Self-Reductions

It is not known how to prove worst-case to average-case equivalence for **NP**-complete problems. In this section we show that an approach similar to the one taken for #**P**, **PSPACE** and **EXP** does not work.

The worst-case to average-case equivalence results that we have seen so far rely on procedures to compute polynomials exactly given a procedure that computes the polynomial well on average. In all cases, given a point x on which we want to evaluate the polynomial, we pick a random line passing through x and then evaluate the good-on-average procedure on points along the line. Since each such point is uniformly distributed, we are able to get the correct value of the polynomial on all points (in the simplest case) or on a majority of points (in the procedures based on Berlekamp-Welch). The following definition captures the way such reductions work.

Definition 4 (Random Self Reduction) *We say that a problem L has a* non-adaptive random self reduction *if there is a probabilistic polynomial time algorithm R such that, given an input $x \in \{0,1\}^n$, $R(x)$ produces a sequence of inputs $y_1, \ldots, y_k \in \{0,1\}^{n'}$, where $k, n' = n^{O(1)}$, and then given $L(y_1), \ldots, L(y_k)$, R decides correctly whether $x \in L$ with high probability. Furthermore, for every $x \in \{0,1\}^n$ and every i, the string y_i is uniformly distributed in $\{0,1\}^{n'}$*

The intuition for the use of a random self-reduction is that if we have an algorithm A that solves L on, say, a fraction $1 - 1/4k$ of the elements of $\{0,1\}^{n'}$, then we can run R, generate $y_1, \ldots, y_k$, and then continue using the answers $A(y_1), \ldots, A(y_k)$, which are going to be all correct with probability at least $3/4$. So A can be used to solve L everywhere on inputs of length n.

We note that the above definition is restrictive in two ways: R generates the points $y_1, \ldots, y_k$ non-adaptively, that is, it chooses y_2 before knowing $L(y_1)$, and so on. Secondly, it is not really necessary that all points $y_1, \ldots, y_k$ be uniformly distributed. Suppose that each point y_i is just generated according to a "smooth" distribution (that is, say, no element of $\{0,1\}^{n'}$ is generated with probability larger than $c/2^{n'}$), and that we have an algorithm A that succeeds on at least a fraction $1 - 1/4ck$ of $\{0,1\}^{n'}$. Then using the randomized reduction together with A we still have at least a probability $3/4$ that all queries are correctly answered.

Nothing is known about the existence of adaptive and/or smooth random self-reduction, but the following theorem rules out non-adaptive uniform random self-reduction.

Theorem 7 *Suppose an* **NP***-complete language L has a random self-reduction as in Definition 4. Then the polynomial hierarchy collapses to Σ_3.*

PROOF: We show that if $L \in$ **NP** and L has a random self-reduction, then the complement of L has an $AM[2]$ protocol (that uses some non-uniformity), and if L were **NP**-complete this would imply the collapse of the hierarchy.

Let R be the random self-reduction for L, let us define a protocol for the complement of L on inputs of length n; let n' be the length of the queries generated by $R(x)$, k the number of queries, and let

$$p = \mathbf{Pr}_{y \in \{0,1\}^{n'}}[y \in L]$$

The value p will be the non-uniformity used in the protocol.

We fix a parameter $m = \Theta((k \log k)^2)$. Given $x \in \{0,1\}^n$:

- The verifier runs $R(x)$ m times indepedently, with fresh randomness each time, and generates queries $(y_1^1, \ldots, y_k^1), \ldots, (y_1^m, \ldots, y_k^m)$. The verifier sends all such queries to the prover.[1]
- The prover responds by telling whether each y_j^i is in L or not, and by certifying all the YES answers with an **NP** witness (remember that L is in **NP**).
- The verifier accepts if and only if the following conditions hold:
  1. $R(x)$ rejects in all the m iterations, assuming the answers from the prover were correct,
  2. all certificates sent by the prover are valid, and
  3. for every $j = 1, \ldots k$ at least $pm - m/2k$ of the queries $y_j^1, \ldots, y_j^m$ are answered YES

We need to argue that if $x \notin L$ then the verifier accepts with high probability when interacting with a valid prover, and that if $x \in L$ then the verifier accepts with low probability when interacting with an arbitrary cheating prover.

If $x \notin L$ and the prover follows the protocol, then $R(x)$ rejects in all the m iterations, and the verifier accepts, provided that the third condition is satisfied. Note that for each fixed j, the strings $y_j^1, \ldots y_j^m$ are independent and uniformly distributed elements of $\{0,1\}^{n'}$, and each one has probability p of being a YES instance. Overall, using Chernoff bounds, there is probability at least $1/4k$ that at least $pm - O(\sqrt{m \log k})$ of them are YES instances, and this number is larger is $pm - m/2k$ if $m \geq c(k \log k)^2$ for a sufficiently large c. Then there is a probability at least $3/4$ that the condition is true for all j.

If $x \in L$, then the verifier rejects if the prover sends valid answers for one of the m runs of $R(x)$, so, if the verifier accepts, it must be because the prover sent a set of answers containing at least m errors. All the erroneous answers of the prover must be YES instances for which it answers NO (if the prover tries to cheat in the other way, it would be caught because of its inability to provide a valid certificate). This means that there is some j such that, among the queries $y_j^1, \ldots y_j^m$, at least m/k are answered NO even though they were YES instances. By the calculation in the above paragraph, very likely there were no more than $pm + O(\sqrt{m \log k})$ yes instances to start with among $y_j^1, \ldots y_j^m$, and so the prover has to pretend that there are only $pm + O(\sqrt{m \log k}) - k/m$, which is less than $pm - k/2m$ by our choice of m, and so the verifier, with high probability, rejects when checking condition (3). $\Box$

[1] To make it an AM protocol, the verifier may just send the randomness it used to generate the m runs of $R(x)$, and let the prover figure out the corresponding queries y_j^i.

Exercises

The following exercises give a proof of Theorem 6.

1. A non-deterministic circuit C has two inputs $x = x_1, \ldots, l_m$ and $y = y_1, \ldots, y_n$. C accepts the string x iff $\exists y. C(x, y) = 1$. Show that every problem in **MA** has non-deterministic circuits of polynomial size. Sligthly harder: show that every problem in **AM**[2] has non-deterministic circuits of polynomial size.

   [*Hint:* The proof is similar to the proof that **BPP** has polynomial sized circuits.]

2. Prove that if **coNP** has polynomial-size non-deterministic circuits, then the polynomial hierarchy collapses.

   [*Hint:* This is not too different from Karp-Lipton.]