6.841 Advanced Complexity	Wednesday, March 19, 2003
Lecture 12	
Lecturer: Madhu Sudan	*Scribe: Nishanth Sastry*

Today:

- IP and AM proofs (Graph Non-isomorphism)
- complexity classes IP, AM, MA etc.
- Some properties:
  - Power of the prover ($IP \subseteq PSPACE$)
  - Goldwasser-Sipser Protocol to prove "largeness"
  - Private coins = Public coins

1 Example Interactive Proof: Graph Non-Isomorphism

Recall from last class that interactive proofs involve a computationally unbounded prover P trying to prove (language membership) to a polytime verifier V. What do interactive proofs look like?

We say two graphs G, H are isomorphic to each other if a relabeling of the vertices of G (i.e., a permutatation of the labels) yields H and vice-versa. Determining whether two graphs are isomorphic is in NP (The certificate is the permutation of the labels). But how do we prove that two graphs are *not* isomorphic?

We can list all permutations of G and prove that none of them will make it equal to H, but this is computationally infeasible. The following shows that graph non-isomorphism admits an interactive proof: Verifier V picks a random graph $F \in \{G, H\}$ and a random permutation of its labels $F' = \pi(F)$ which it sends to the prover P. V accepts if P can guess whether F' came from G or H. If G, H are indeed non-isomorphic, P can correctly guess the original graph of F since it is assumed to be computationally unbounded. Otherwise, F, G and H are isomorphic to each other and even P cannot distinguish them apart (At best it can guess correctly with a probability of 1/2.).

2 Complexity classes: Interactive Proofs and Arthur-Merlin Games

The idea of proofs by interaction was independently developed by Goldwasser, Micali and Rackoff who called it "Interactive Proofs" and by Babai and Moran who called it "Arthur-Merlin games". In current notation, we use IP to denote all languages with poly-round interactive proofs. In Arthur-Merlin games, Arthur is a polynomial-time bounded entity analogous to the Verifier in IP and Merlin is capable of unbounded computational power, just like the Prover in IP. Arthur is allowed to toss coins and challenge Merlin but the key difference from IP is that the coins tossed are public. We denote 1-round games where Arthur goes first and Merlin responds by AM. Similarly MA denotes 1-round games where Merlin goes first and Arthur responds. We can extend this notation to AMAMA and so on. For convenience, we will also use the notation IP[k] to denote an interactive proof (private coins) with k rounds of interaction. Similarly define AM[k] and MA[k].

Some of the factors that could affect the computational power of interactive proofs are:

- Interaction: Number of rounds (constant/polynomial)

- Verifier's Randomness: whether the coin tosses are made public kept private.
- Error: One-sided error vs. two sided error

In the rest of the lecture, we will focus on some of these aspects.

3 Power of the prover

For any fixed verifier, what should the prover do to maximize the probability of acceptance? Consider the quantity $p_{q_1 a_1 \ldots q_i}$, which denotes the max probability at round i over all future answers that verifier accepts. This is simply the probability that the verifier accepts if the prover gives the best possible answer:

$$p_{q_1 a_1 \ldots q_i} = \max_{a_i} \{p_{q_1 a_1 \ldots q_i a_i}\}$$

We can compute this recursively in terms of all possible questions q_{i+1} that can be asked by the verifier in round $i+1$.

$$p_{q_1 a_1 \ldots q_i a_i} = \sum_{q_{i+1}} Pr[q_{i+1}|\text{past}] p_{q_1 a_1 \ldots q_i a_i q_{i+1}}$$

Since the prover is unbounded computationally, it can compute this by induction over the number of remaining rounds. Since this computation can be done in PSPACE, we have the following theorem:

Theorem 1 $IP \subseteq PSPACE$

In a later lecture, it will be shown that IP = PSPACE.

4 Amplification: sequential and parallel (round-preserving)

The IP verifier was defined to accept good strings with a probability of at least 2/3 and bad strings could be accepted with a probability of no more than 1/3. Analogous to the amplification lemma for the randomized complexity classes, we can reduce the error to any polynomially small value by repeating the interactive proof over many trials and accepting only if a majority of the trials accept. However, notice that the questions asked in each round in a given trial depend only on the previous round(s) of that trial. Thus, the questions asked in round i of each trial are independent of each other and we might as well run the trials in parallel. Thus, we can amplify the error while preserving the number of trials.

5 Private coins are equivalent to public coins

It seems that IP should be more powerful than the arthur-merlin model because the coin-flips made by the verifier are secret whereas Arthur has no secrets from Merlin. However, we can convert one to the other: Consider an IP proof. Suppose Arthur asks Merlin to prove that there are a large number of random strings on which to accept. If Merlin can successfully show this, then IP reduces to Arthur-Merlin games, because regardless of the random coins flipped, Merlin can make Arthur accept with high probability.

5.1 Goldwasser-Sipser Protocol

This is a protocol that demonstrates the largeness of sets by approximate counting. We will use this as a "subroutine" in proving the equivalence of IP[k] and AM[k]. Consider a problem defined by the following inputs: $S \subseteq \{0,1\}^n$, and $m = O(\sqrt{n})$. We would like the verifier to accept with high

probability if S is BIG, defined as $|S| > 2^m$ and reject (accept with probability less than a third) if S is SMALL, defined as $|S| < \frac{2^m}{100}$. Furthermore, it is guaranteed that S is either BIG or SMALL.

The solution is similar to the Valiant-Vazirani reduction from a previous class. The idea is to use a hash function to map $\{0,1\}^m$ to a sufficiently small space, say $(H) = \{0,1\}^{m-4}$.If S is large enough, then no matter which $y \in (H)$ we pick; we can find an element in S that maps to it.

Claim 2 *If $|S| > 2^m$, and h is chosen from a family of pairwise independent hash functions, then for at least $2/3$ of the y's, $\exists x \in S$ such that $h(x) = y$.*

Claim 3 *If $|S| < \frac{2^m}{100}$, for any hash function h at most $16/100 < 1/6$ of the y's have $x \in S$ such that $h(x) = y$.*

Thus the following protocol verifies that S is large: Merlin sends a random hash function $h : \{0,1\}^m \to \{0,1\}^{m-4}$ to the Arthur. Arthur sends a random $y \in \{0,1\}^{m-4}$ back to Merlin. If S is large enough, then Merlin is able to find $x \in S$ such that $h(x) = y$ to send to Arthur. Otherwise, Merlin will succeed in finding a suitable x with a probability less than a third (actually, less than $1/6$) as required.

Notice that Arthur does not even need to be able to verify that $x \in S$ efficiently (i.e., we do not need $S \in BPP$). As long as $S \in IP$, we can verify the largeness of S by continuing the above protocol by executing the protocol that verifies membership in S!

5.2 1-round protocols: equivalence of public and private coins

We will show that a 1-round IP protocol can be simulated by an Arthur-Merlin game with a constant number of rounds. This can be extended to any number of rounds of IP.

The goal is to provide an AM[$O(1)$] protocol to decide an arbitrary IP[1] protocol. We consider what an "optimal" prover would be able to do in a given round. The prover should be able to commit to an answer from the question asked such that for a large number of (private) random coin tosses in the future, this answer will lead the verifier to accept.

We first fix a verifier V with $2/3$ completeness and a soundness $1/\text{poly}$. Let Q be the set of questions V can ask. For each $q \in Q$, let S_q be the set of (private) random strings that lead to q being asked. Let r be the length of the random strings. If the IP[1] protocol accepts an input, we want our AM protocol to accept on at least $(2/3)2^r$ random strings. Otherwise, we want it to accept over a very small number of strings, say $1/\text{poly}(r)$. In other words,

$$\sum_{q \in Q} |S_q| \begin{cases} \geq (2/3)2^r & \text{if verifier accepts} \\ \leq 1/\text{poly}(r) & \text{if verifier rejects} \end{cases}$$

For simplicity assume that some of the questions are never asked, and for all other questions, exactly 2^l random strings lead to that question: $|S_q| = 0$ or $|S_q| = 2^l$. For this we need at least $(2/3)2^{r-l}$ questions for which prover has an answer that the verifier will accept. To show this, we use the Goldwasser-Sipser protocol: Merlin sends a hash funchtion h; Arthur responds with a y; Merlin replies with $q \in Q$ such that $h(q) = y$. To complete the proof, Arthur needs to verify that $|S_q| \geq 2^l$. This is accomplished by another Goldwasser-Sipser protocol (with q, now known to both sides, as input).

6 $AM[k] = AM$

The idea is that Arthur corresponds to the BP operator; Merlin corresponds to the $\exists$ operator. Thus $\text{AMAM}(x) = \text{BP}\exists\text{BP}\exists(x)$. We can exchange the BP and $\exists$ operators as in the proof of Toda's theorem.