18.405J/6.841J Advanced Complexity Theory	April 13, 1999
Lecture 17	
Lecturer: Daniel A. Spielman	*Scribe: Navid Sabbaghi*

Today we study a new topic: Interactive Proofs. Interactive proofs were motivated by the question: what models allow us to be convinced of membership of a word in a language with high probablity? Interactive proofs are a model for achieving that conviction through communication rather than the mere presentation of a proof. In this lecture we will be interested in seeing what the power of this model is (what classes of languages that we are already familiar with, are included in it?).

1 Interactive Proofs

So what are Interactive proofs? Imagine two parties and the conversation between them. One party is an "all-powerful" *Prover*, which we denote by P. The Prover is a prodigy and has unlimited computational resources and more formally can be defined by an arbitrary function. The second party is the *Verifier,* which we denote by V. The Verifier is a polynomial time Turing Machine, which has access to randomness in the sense that it can toss coins and ask different questions depending on the outcome. The coins are kept *private* to the Verifier. The conversation is a sequence of "questions" from V and "answers" from P in which the Prover is trying to convince the Verifier of some statement by answering the Verifier's questions. The questions and answers are restriced to being polynomial in the length of the input.

Why do we choose these constraints? Does this model make sense? Well it is essential that V has access to randomness because otherwise we will never escape NP. In the absence of randomness available to V, P can predict exactly what V will ask and precompute the whole sequence of questions and answers, so in fact the interaction would not be needed. If V can toss coins, however, P cannot predict what the question is going to be.

What are some examples of this protocol? A simple example of an interactive proof is a protocol for convincing a colorblind friend that colors exist. The protocol is: you, the Prover, get two balls of different colors (say red and blue) which are otherwise identical. You give the balls to your colorblind friend, the Verifier, and tell him that he has a red ball in his right hand and a blue ball in his left hand. The friend, the Verifier, then hides the balls behind his back and randomly shuffles them, so that he knows which ball is in which hand but you don't. He then shows them to you and you correctly identify the balls. Therefore the friend, the Verifier, will be convinced of the existence of colors with high probability. Graph Non-Isomorphism is another problem that is can be solved with an interactive proof.

1.1 Interactive Proof Example: Graph Non-Isomorphism

We first define Isomorphism of graphs: two graphs G and H are isomorphic if one is a permutation of the other. More formally,

Definition 1 *Two graphs G and H are* ***isomorphic*** *if*

- *G and H have the same number of nodes n,*
- *$\exists$ a permutation π of vertices of graph G such that edge $(\pi(i),\pi(j)) \in G \iff$ edge $(i,j) \in H$.*

Now proving that graphs are isomorphic is easy. A "proof" of isomorphism is simply the permutation, π. But think about how one would prove that two graphs are *not* isomorphic. That problem has been approached deterministically using ideas from group theory and the best time bound so far for the general graph non-isomorphism problem is TIME($2^{O(\sqrt{n})}$). If we restrict the class of graphs to say bounded degree graphs, polynomial algorithms do exist, but still they are very far from trivial. At any rate, we will see that there is a simple Interactive Proof for the graph nonisomorphism problem.

The simple interactive proof is very similar to convincing a colorblind friend that colors exists. Say you (the friend, verifier) are given two graphs G and H. You can put them behind your back and mix them up and pull one out and ask the prover "is this graph the one that was originally in my left or right hand?" If they are indeed isomorphic (ie the same graph) then the prover should not be able to tell which one is which.

More formally, we define $NONISO$ as a language of all pairs of graphs (G, H) such that G and H are **not** isomorphic. We now give an interactive proof for $NONISO$. We specify the protocol by which P will convince V that two graphs G and H are non-isomorphic. We assume that G and H have the same number of vertices, since otherwise the graphs are obviously non-isomorphic. Let n denote the number of vertices of either graph.

The IP Protocol

1. VERIFIER: Flip a random bit $b \in \{0, 1\}$; b is a "fair" coin with probability 1/2 of either outcome. This will decide which graph the verifier will "play" with.
2. VERIFIER: Choose a random permutation π of the verticies $\{1, 2, ...n\}$. If $b = 0$ show the Prover $\pi(G)$. Otherwise show $\pi(H)$
3. PROVER: In response the Prover says which graph is being shown to it.
4. VERIFIER: If the prover is correct, then accept, else reject.

This can be repeated as many times as needed to boost confidence. As long as the prover guesses correctly, the verifier should become more confident that graphs are non-isomorphic.

Proof of Correctness.

We use notation (V, P) to denote the outcome of the protoocol.

If G is not isomorphic to H then there exists a prover P s.t. $Pr[(V, P) \text{ accept}] = 1$

If G is isomorphic to H, then for any P, on k runs of the protocol $Pr[(V, P) \text{ accept}] \leq \frac{1}{2^k}$. The latter is because P cannot distinguish between two isomorphic graphs.

1.2 Formal definitions

So now in this section we will formalize the notion of an Interactive Proof. In general, the key elements of an Interactive Proof are

1. players V and P take turns
2. after a certain number of turns they stop and V decides whether to accept or reject.

We can define V as a function $V(w, r, C) \to \{S, accept, reject\}$, where w is an input string, r is a random string and C and S are strings denoting respectively the conversation so far and V's response. More specifically, the output of this function is defined as follows:

if $C = \varepsilon$ output $S = V_1$

if $C = V_1, P_1, ...V_i, P_i$ and $i < k$ (where k is the chosen total number of rounds of the protocol) then output V_{i+1}

if $C = V_1, P_1, ...V_k, P_k$ output *accept* or *reject*

Similarly, we can define P as a function $P(w, C) \to S$, where S is P's response string. If $C = V_1, P_1, ...V_i$, then P outputs P_i.

Now we can define class $IP(k)$ as a class of all languages that have interactive proofs consisting of k rounds. We say that a language $L \in IP(k)$ if there exists a polynomial-time Turing Machine V that rejects or accepts after k rounds s.t.

- $w \in L \Longrightarrow \exists P$ s.t. $Prob_r[(V(w, r), P(w)) \text{ accept}] \geq 2/3$
- $w \notin L \Longrightarrow \forall P \; Prob_r[(V(w, r), P(w)) \text{ accept}] \leq 1/3$

The IP model is sometimes called a "private coins" model because the prover does not get the coin-flips r of the verifier as input. We will discuss "public coins" later in the lecture and in the next lecture show that interactive proofs with private coins are equivalent to interactive proofs with public coins. But first, we derive a useful consequence of this equivalence.

1.3 Some useful results about IP

Using the "public coins = private coins" result, a speedup theorem for constant-round interactive proofs that we will derive, and a problem from the homework, one can show that that for any k, IP(k) $= BP \cdot \sum \cdot P$ and that $BP \cdot \sum \cdot P \subseteq$ NP/*poly*. From these inclusions, we can conclude that graph isomorphism, ISO, is probably not NP-complete.

Sketch of Proof Because if ISO were NP-complete that would imply that NONISO is coNP-complete. But we have an Interactive Protocol for graph NONISO and all constant round IPs can be done in $BP \cdot \sum \cdot P$. This implies that $CoNP \subseteq BP \cdot \sum \cdot P \subseteq$ NP/*poly*, which implies that that the polynomial hierarchy collapses which at this point people believe is unlikely. ■

2 Arthur Merlin Games aka Public IP

Are there generalizations of the Interactive Proof model? Note one generalization of this model would be to make the outcome of the coins that the verifier tosses *public*. This small change leads to a model of Proofs called Arthur-Merlin games. We will see that the "public" proof model, ie Arthur-Merlin games, and the "private" proof model are equivalent.(Sipser and Goldwasser) and anything you can do in IP without access to random bits you can do with access to random bits.

In Arthur-Merlin games the idea is that Arthur (the verifier) flips random coins at each round, but unlike the private coins model, the coins are visible to Merlin (the prover) as well. Merlin then tries to persuade Arthur of the answer. Note that it is important that the coins are tossed at each round, rather than all at once, because otherwise we can't escape $BP \cdot \sum \cdot P$ (for the same reason as the fact that if we had no randmoness at all in interactive proofs, we wouldn't escape NP). Note also that since Merlin is all-powerful, he does not need to see anything but the coin-tosses of Arthur, because once Merlin knows the coin tosses, he can compute what Arthur is going to ask.

The behavior of the Verifier $V(w, C)$ (Arthur) can then be described as follows:

if all k rounds are done V decides whether to accept or reject

if less then k rounds are done, V outputs some random bits

This leads to complexity classes $AM(k)$ and $MA(k)$, where k is the number of rounds and the order of letters A and M is determined by who (Arthur or Merlin) starts the conversation. We will use AM for $AM(1)$ and MA for $MA(1)$ (this notation will become clear once we prove one of the theorems stated below).

2.1 Useful Theorems about Arthur-Merlin Games

It turns out that

$\sum \cdot BP \cdot P = MA$

$BP \cdot \sum \cdot P = AM$

However it is not clear how $BP \cdot \sum \cdot BP \cdot \sum \cdot P$ is related to AM(k).

On the problem set you will show that MA $\subseteq$ AM. We will later prove several theorems:

Theorem 2 *$AM(k) = AM$ for all constants k*

This means that nothing is gained by having a constant number of rounds compared to a single round.

Theorem 3 $AM(f(n)) = AM(\frac{f(n)}{2} + 1)$

This means that it is always possible to speedup the protocol by a factor of 2. You can apply this theorem recursively, but only a constant number of times, because otherwise the size of each message between Arthur and Merlin blows up exponentially.

Theorem 4 $AM(f(n)) = IP(f(n))$

This means that private coins are equivalent to public coins