6.841 Advanced Complexity Theory	Apr 4, 2007
Lecture 15	
Lecturer: Madhu Sudan	*Scribe: David Glasser*

The classes AM and IP. Proof by Goldwasser and Sipser that private coins with 2-sided error can be simulated with public coins and 1-sided error. Start to show that PSPACE is in IP.

1 AM and IP

When defining notions of interactive proof protocols, there are three aspects of the definition that appear flexible. First, the verifier's coin tosses can either be public (visible to the prover) or private (hidden from the prover). Second, the error rate can be one-sided (always accepting if $x \in L$, say) or two-sided. Third, the bound on the number of rounds of interaction can be constant or polynomial. We showed last time (with the proof of Kilian) that for polynomially many rounds, the same languages have proofs with private coins as with public coins, and with one-sided error as with two-sided error. (Specifically, we converted a two-sided error private coin protocol into a one-sided error public coin protocol). Today we will show that the same is true for languages with a constant number of rounds. However, we do believe (but can't show) that the number of rounds *is* important. For somewhat-historical reasons, we call the class of languages with constant-round protocols AM (after Arthur-Merlin protocols) and the class of languages with polynomial-round protocols IP (after interactive proofs); we believe that AM is strictly contained in IP.

The standard two-rounds public-coin Arthur-Merlin model for AM is as follows. L is in AM if there exists a PPT V such that $x \in L$ implies that the probability over random strings R that there exists an a such that $V(x, R, a)$ accepts is at least $c(n)$, and if $x \notin L$ then the probability is at most $s(n)$. For two-sided error, we can set $c(n) = \frac{2}{3}$ and $s(n) = \frac{1}{3}$, though the exact values don't matter (as with BPP). Recalling the quantifiers for complexity classes introduced in the proof of Toda's theorem, we can also define AM as $BP \cdot \exists \cdot P$. Now, $AM[k]$ (AM with k rounds of interaction) is $BP \cdot \exists BP \cdot \exists \cdots BP \cdot \exists \cdot P$, but via a proof similar to Toda's theorem, we can show that $BP \cdot \exists \cdot \mathcal{C} = \exists \cdot BP \cdot \mathcal{C}$; thus any constant number of rounds of interaction (starting with Arthur) gives us the class AM. (The proof that this swap works is on problem set 3.)

2 Goldwasser-Sipser Approximate Counting Protocol

We haven't proved yet that coin privacy and 1-sided error is irrelevant for constant-round protocols. In order to do so, we examine the problem of approximating the size of a set. (We will eventually apply this to a set of coin tosses which make a protocol accept.)

The input to this promise problem is a set $S \subseteq \{0,1\}^n$ and a number N. S is specified not explicitly but by a protocol proving membership. Because we are using this in the context of interactive proofs, this protocol can be in AM. The problem is a YES example if $|S| \geq N$ and NO if $|S| < \frac{N}{10n^2}$ (the precise formula here doesn't matter, but the point is that there is a reasonable gap where we aren't expected to be able to determine the answer). We'll come up with an Arthur-Merlin protocol for this; we'll first consider some special cases. (Note first that this is only hard for N which is superpolynomial in n; for small N, Merlin can just provide proofs that N specific elements are in S.)

What if $N = 2^n$? That is, it is a YES instance if every element of $\{0,1\}^n$ is in S, and a NO instance if an element is in S with probability at most $\frac{1}{10n^2}$. We can accomplish this pretty easily with one-sided error: the verifier (Arthur) picks a random $x \in \{0,1\}^n$ and asks the prover to prove that $x \in S$. The protocol is accepted iff the prover manages to do so. If this is a YES instance, the prover can definitely prove $x \in S$; if this is a NO instance then the prover can only prove $x \in S$ (and make the protocol incorrectly accept) with probability $\frac{1}{10n^2}$. Thus this is an AM protocol for the special case $N = 2^n$.

What if N is still big, but not the whole universe? Say, $N = \frac{2^n}{100}$. The *prover* starts the protocol by stating $O(n)$ strings $x_i \in \{0,1\}^n$; the verifier picks an $x \in \{0,1\}^n$ at random and sends it; the prover then

proves that for some i, $x \oplus x_i \in S$. The idea is that if S is pretty big, there exists some tuple $(x_1, \ldots, x_{O(n)})$ such that any fixed "shift" of its elements will leave at least one element in S; but if S is a factor of n^2 smaller, it is likely that a random shift will leave all of the elements out of S. The proof that this works (with one-sided error!) is very similar to the Lautemann-Sipser proof that BPP is in Σ_2^P. This sort of protocol works for any $N = \frac{2^n}{poly(n)}$ but stops working once S is small enough that the prover can't find a tuple whose shifts all intersect with S. For example, this doesn't work for $N = 2^{\sqrt{n}}$.

So in the most general case, the problem is that S may be much smaller than $\{0,1\}^n$. Thus it's really hard for us to "guess" elements in S. The trick we use is the same as in Valiant-Vazirani: hashing! If we can come up with a "good" hash function h mapping us into a smaller universe then we're mostly reduced to a previous case: showing that $h(S)$ fills up a large part of its universe. The verifier chooses an $m = O(\log N)$ and a hash function $h: \{0,1\}^n \to \{0,1\}^m$ and sends them to the prover; we recall from the Valiant-Vazirani proof that there exist universal hash function families where a randomly chosen function is "good". Then just like in the previous special case, the prover sends l elements $y_i \in \{0,1\}^m$. The verifier picks a $y \in \{0,1\}^m$, and then the prover sends back an i and an $x \in \{0,1\}^n$ such that $h(x) = y \oplus y_i$ and $x \in S$. Combining some calculations from the Lautemann-Sipser and Valiant-Vazirani proofs, we can show that this is in fact an AM protocol for the set size approximation problem.

3 Applying Goldwasser-Sipser to public vs private coins

Now let's use that problem to show that (for constant rounds) public and private coins are equivalent. What does a general one-round private-coin protocol look like? V chooses a random $R \in \{0,1\}^n$, computes a question $q = q(x, R) \in \{0,1\}^l$, and sends it to P; P responds with some a. If $V(x, R, a)$ accepts, then we accept x. If x is in the language, then the set of R making V accept should be large; if it's not, then it should be small. Thus we should be able to somehow apply the Goldwasser-Sipser problem...

Fix q and a; let S_{qa} be the set of R with $q(R) = q$ and $V(x, R, a)$ accepts. (For example, consider graph non-isomorphism; here q is a graph (a permutation of either G_1 or G_2 and a is both a statement of which graph it's isomorphic to *and* a proof of that isomorphism.)

Let's consider a simple case first. For $x \in L$, then for all q there exists a such that $|S_{qa}| \geq N$; for $x \notin L$, then for all q and a, $|S_{qa}| \leq \frac{N}{10n^2}$. Additionally, we assume that somehow the verifier knows (and believes!) N. We can use a pretty simple protocol here: first P says a q and an a, and then it goes into a Goldwasser-Sipser protocol to prove that $|S_{qa}| \geq \frac{2}{3}N$. That is, it says "here's a possible question/answer pair which would have made you accept; I'm going to prove that it is likely you would have asked this question". However, in general N depends on q (we'll call it N_q), and the verifier doesn't actually know its value.

How do we get around this? The prover will tell us a "typical" value of N_q, which it will call N, and it will prove that lots of questions have $N_q \geq N$. Specifically, it will prove that $|\{q \mid N_q \geq N\}| \geq \frac{2^n}{N} \cdot \frac{1}{2n}$. How will it do that? By using Goldwasser-Sipser!

So the complete protocol looks like this. The prover sends a value N to the verifier. It then starts a Goldwasser-Sipser protocol to prove that $|\{q \mid N_q \geq N\}| \geq \frac{2^n}{N} \cdot \frac{1}{2n}$. Then it chooses a specific q_0 and proves that $N_{q_0} \geq N$ with another Goldwasser-Sipser protocol.

4 Next Time: IP is PSPACE

We already know that IP is contained in PSPACE. Over the next two lectures, we will show that the converse is true as well. Here's some intuition and history. Lipton (1989) showed that the permanent is "random self-reducible". (Recall that the permanent of a matrix is like the determinant, without the alternating signs. It's also the number of cycle covers of a graph. It's hard to compute!) That is, if there is an efficient algorithm that can compute the permanent of random matrices then there is an efficient algorithm to compute the permanent of any matrix. I.e., a good algorithm for the average case implies a good algorithm for the worst case.

Next, Blum-Luby-Rubinfeld (1990) showed "checkability" (that you can check that a program is always correct — it's a promise sort of thing, so you can check that either the instance you gave multiplied correctly, or that we can find some example where the algorithm is wrong) is implied by random self-reductions and

shrinking self-reductions. Now permanent is self-reducible: the standard calculation based on minors is a shrinking self-reduction. So the permanent is checkable. Then Lund-Fortnow-Karloff-Nisan showed that this means that permanent is in IP. This was the first proof that coNP is in IP. In fact, the whole hierarchy is in IP, because PSPACE is. We will start to show this next time.