18.405J/6.841J Advanced Complexity Theory	April 30, 1998
Lecture 21	
Lecturer: Daniel A. Spielman	*Scribe: Yoav Yerushalmi*

In this lecture, we will prove the following two things:

- $\#P \subseteq IP$
- $IP(poly) = PSPACE$

The former result is only used as a basis for seeing how to prove the latter. However, the fact that an interactive protocol with a polynomial number of rounds is equivalent to a machine running in polynomial space was a surprising result.

In this lecture, we will only actually prove this for public coins (that is we will show that $AM(poly) = PSPACE$).

1 $\#P \subseteq IP$

This fact was proven using arithmetization. The idea behind arithmetization is to convert the problem (claim/language) into a mathematical problem, and then use well established mathematical formulas to reduce or modify the problem to one that can be solved by our machine.

Example 1 (Arithmetization of 3CNF) *Given a 3CNF, we let the variables in the 3CNF formula be variables of a polynomial, where if a variable's value in the polynomial is 0 it means the variable is* **false** *in the 3CNF, and vice-versa.*

$$\begin{aligned} not(x_i) &= 1 - x_i \\ x_1 \wedge x_2 &= x_1 \cdot x_2 \\ x_1 \vee x_2 &= 1 - (1 - x_1)(1 - x_2) \\ (x_1 \vee x_2 \vee \overline{x_3}) &= 1 - (1 - x_1)(1 - x_2)(x_3) \end{aligned}$$

The second to last formula was derived by DeMorgan's law. Each of thee clauses in the 3CNF lead to a degree 3 polynomial. All these polynomials are multiplied (AND) together. The polynomial can equal 1 if and only if the entire equation had a satisfying assignment.

The reason we use this polynomial, however, is that it can take inputs in a domain OTHER than $\{0, 1\}$. So, now say we want to convince a verifier that we know the number of satisfying assignments toa 3CNF. This is analogous to making the claim that we know the value of the following function:

$$F() = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(x_1, x_2, \cdots, x_n)$$

The problem is that the verifier doesn't have the required amount of time complexity to compute the above function. Our trick will be to use the prover's extra power to convince the verifier that the value it claims for the above equation is true by convincing him that an equation that is directly related to the above, but that the verifier CAN compute, is correct.

If P can convince V that the value for $F()$ it has provided is the correct number of satisfying assignments to the 3CNF formula, then we know that $\#P \subseteq IP$. In order to do this, we need a way to reduce the equation above. We use a construct named ϵ-links to do that.

Definition 1 (ϵ-link) *An ϵ-link is an IP(poly) protocol that takes a proposition A to another proposition B. That is:*

- *if A is true, then $\exists P$ prover such that the proposition B is also true.*
- *if A is false, then $\forall P$ provers, the probability that B is true and the verifier V fails to detect a deception is $\leq \epsilon$.*

Ideally, we only care for ϵ-links that simplify A. If we have an ϵ-link for A, we can use B to convince V with probability $\geq 1-\epsilon$.

Our proof of $\#P \subseteq IP$ will involve a chain of links that take the difficult 3CNF polynomial to a simple one that V can perform to convince itself that P has computed $\#P$. We will have an error bound which is the sum of all the errors we encounter (one per link), so we will need to set our ϵ to be small.

Our first ϵ-link will take our original claim:

$$A: \quad \left[\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(x_1, x_2, \cdots, x_n)\right] = C$$

to a new claim B using the following protocol:

1. $P \longrightarrow V$ a polynomial of one variable $q_1()$ which will be directly related to the x_1 value thus:
$$q_1(x_1) = \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(x_1, x_2, \cdots, x_n)$$
2. V checks that $q_1(0) + q_1(1) = C$. If this is not the case, we know that the polynomial was not computed correctly or the value we were given for C was wrong. Therefore, we reject.
3. V then chooses an α at random from a field which we shall name $\mathbb{F}$ and decide its size to adjust for ϵ (computation for its size will be done later). V tells α to P, and they now have a simpler equation to work with:
$$B: \quad \left[\sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(\alpha, x_2, \cdots, x_n)\right] = q_1(\alpha)$$

Claim 1 *The above protocol is an ϵ-link from A to B.*

Proof

1. if A is true, and P follows the protocol, B is clearly true.
2. if A is not true, then B is very likely to be false: Let $N = |\mathbb{F}|$ (choose N prime). If A is false, then
$$\Pr_{\alpha}\left[q_1(\alpha) = \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(\alpha, x_2, \cdots, x_n)\right] \leq \frac{deg(p)}{N}$$
(We know that $deg(p) \geq (q_1)$). We set $\epsilon = \frac{deg(p)}{N}$, and voila.

■

We now have a polynomial that has one less variable (x_1 has been replaced by a more complex polynomial that absorbs α). We are one step closer to making it possible for V to compute $p()$. We now have to repeat this idea to reduce the polynomial by eliminating one variable at a time. This leaves us with a remaining proposition:

$$p(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) = C_n$$

Which the verifier can easily check, although we have now introduced n error terms.

If the verifier accepts, the probability that the initial claim was false is at most $n \cdot \epsilon = n \cdot \frac{deg(p)}{N}$. So, we simply set $N \gg n \cdot deg(p)$ to ensure a low probability. if N is exponential, we get $< 1/poly()$ error.

We have now shown that we can simulate $\#P$ using IP where P knows the solution, so we've shown that $\#P \subseteq IP(poly)$.

2 $IP(poly) = PSPACE$

Now we get onto the impressive result, using stuff we built on in the previous section. We demonstrate this fact using the standard technique of showing that $IP \subseteq PSPACE$ and that $IP \supseteq PSPACE$.

2.1 $IP(poly) \subseteq PSPACE$

This is certainly the easier one of the pair. In fact, it's intuitive. We can simulate the IP protocol using the $PSPACE$ machine.

2.2 $IP(poly) \supseteq PSPACE$

We earlier showed that, given $p(x_1, x_2, \ldots, x_n)$ of polynomial degree, there is an ϵ-link from the assertion that

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} p(x_1, x_2, \cdots, x_n) = C$$

to

$$p(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n) = C_n \quad \text{with } 1/poly() \text{ error}$$

this time we're going to have polynomials V can't verify, but P will convince V of value of polynomial. Recall the proof from an earlier lecture that Alternating P-time equals PSPACE. We will follow a similar line:

- begin with a PSPACE machine M which works on input w.
- associate with it a graph. The nodes of the graph correspond to possible (head,tape,state) configurations of the machine, and the edges correspond to legal transitions.
- accept if and only if there is a path from the start node to an accepting node (in this case, there is only one such path since the machines are both deterministic).

We then invented a function named $FromTo(x, y, m)$, which was true if you could go from node x to node y in $\leq m$ steps. And we made assertions such as

$$FromTo(a, b, i) = \exists_z \left(FromTo(a, z, \lfloor i/2 \rfloor) \wedge FromTo(z, b, \lceil i/2 \rceil) \right)$$

Our base case: $FromTo(x, y, 1)$ can be evaluated in polynomial time. Also, $FromTo(x, y, 1)$ can be evaluated by a 3CNF, which can be evaluated by a polynomial of degree poly(n). It variables are the bits of x and y.

The idea is to start from the original claim $FromTo(x, y, j)$ (using m bit strings to represent x, y, j, and reduce it to a multiplication (and) of lots of $FromTo$'s os size one using the following principle:

$$FromTo(x, y, j) = \sum_{z_1 z_2 z_3 \cdots z_m} FromTo(x, z, j/2) \cdot FromTo(z, y, j/2)$$

This idea will be explored further and proved in the next lecture.