PRINCETON UNIVERSITY COS 522: COMPUTATIONAL COMPLEXITY

Lecture 12: Interactive Proofs

Lecturer: *Sanjeev Arora* Scribe: *Carl Kingsford*

Recall the certificate definition of **NP**. We can think of this characterization of **NP** as an interaction between two entities P and V. For a language $L \in \mathbf{NP}$, x is in L if and only if P can send V a certificate (that depends on x) that V can use to check in polynomial time that x is indeed in L. P is the "prover" and V is the "verifier." For **NP** only one interaction is allowed. We can expand the power of the verifier TM: let it be a *probabilistic* TM that can make a *polynomial number* of queries of the prover. Now the verifier and the prover exchange a polynomial number of polynomial length messages. Both the prover and verifier see the input x. The prover is trying to convince the verifier that $x \in L$. Formally, we define the class **IP** of languages that have such interactive proofs:

Definition 1 (**IP**) *A language L is in* **IP** *if there is probabilistic, polynomial-time TM V with coin flips r that interacts with an all-powerful prover, where in round i its query $q_i(x, r, a_1, \ldots, a_{i-1})$ depends on the input, the random string, and the prover's responses $a_1, \ldots, a_{i-1}$ in the previous rounds. The verifier has the property that:*

(i) $x \in L \Rightarrow \exists P \quad \Pr_r[V \text{ accepts } x \text{ after interaction with } P] \geq 2/3$

(ii) $x \notin L \Rightarrow \forall P \quad \Pr_r[V \text{ rejects } x \text{ after interacting with } P] \geq 2/3.$

Since P cannot see the coin flips of V, we say the protocol is private coin. *We further define* $\mathbf{IP}[k]$ *(for $k \geq 2$) be the set of all languages that have a k round interactive proof in this private coin model, where a "round" is either a query or a response.*

If V were not allowed to be probabilistic it is easy to see that **IP** is equivalent to **NP**: the prover can compute all the queries the verifier will make ahead of time and offer this entire transcript to the verifier straightaway. By allowing V random bits, we get a more powerful class.

The probabilities of correctly classifying an input can be made arbitrarily large by using the same boosting technique we used for **BPP**: sequentially repeat the protocol k times. If $x \in L$ then the same prover can be used on each repetition. If $x \notin L$ then on each repetition, the chance that the verifier will accept x is less than 1/3.

We can define similar classes **AM** and **AM**$[k]$ in which P *does* see the coin flips of V—this is the *public coin model.* We state the following comments about **IP**$[\cdot]$, **AM**$[\cdot]$, without proof:

(i) **IP**$[k] \subseteq$ **AM**$[k+2]$ for all constants k.

(ii) For constants $k \geq 2$ we have **AM**$[k] =$ **AM**$[2]$. This is surprising because **AM**$[k]$ seems similar to **PH** with the $\forall$ quantifiers changed to "probabilistic $\forall$" quantifiers, where *most* of the branches lead to acceptance. See figure 1.

It is open whether there is any nice characterization of **AM**$[\sigma(n)]$, where $\sigma(n)$ is a suitably slow growing function of n, such as $\log \log n$.

(iii) Changing 2/3 to 1 in the definition of **IP** does not change the class. That is, defining the **IP** in a manner similar to **coRP** is equivalent to defining it like **BPP**.

Whereas **BPP** is a probabilistic version of **P**, **AM**[2] is as probabilistic version of **NP**. **AM**[2] can be thought of as "**BP-NP**"—languages for which there is a bounded probabilistic *nondeterministic* TM.

(iv) It is relatively easy to see that **AM**$[poly(n)] \subseteq$ **PSPACE**; the proof is left as an exercise. It is possible, in **PSPACE**, to come up with a good strategy for choosing the $\exists$ edges in figure 1.

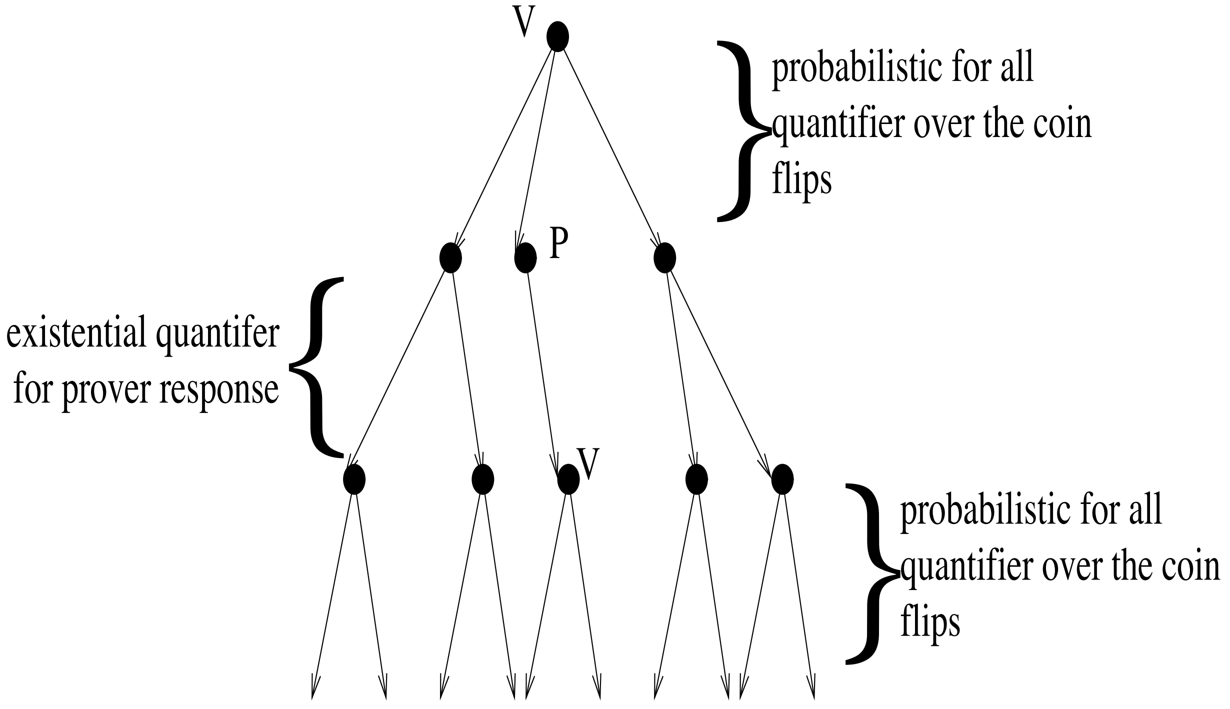
V
probabilistic for all quantifier over the coin flips
P
existential quantifer for prover response
V
probabilistic for all quantifier over the coin flips

Figure 1: **AM**$[k]$ looks like $\prod_k^p$

EXAMPLE 1 (GRAPH NON-ISOMORPHISM) Given two graphs G_1 and G_2 we say they are isomorphic to each other if there is a permutation π of the labels of the nodes of G_1 such that $\pi G_1 = G_2$. The graphs in figure 2, for example, are isomorphic with $\pi = (12)(3654)$. If G_1 and G_2 are isomorphic, we write $G_1 \equiv G_2$. The GRAPH NON-ISOMORPHISM problem is this: given two graphs, are they *not* isomorphic?

It is clear that the complement of GRAPH NON-ISOMORPHISM is in **NP**— a certificate is simply the permutation π that is an isomorphism. What is more surprising is that GRAPH NON-ISOMORPHISM is in **IP**. To show this, we give a private-coin protocol that satisfies definition 1:

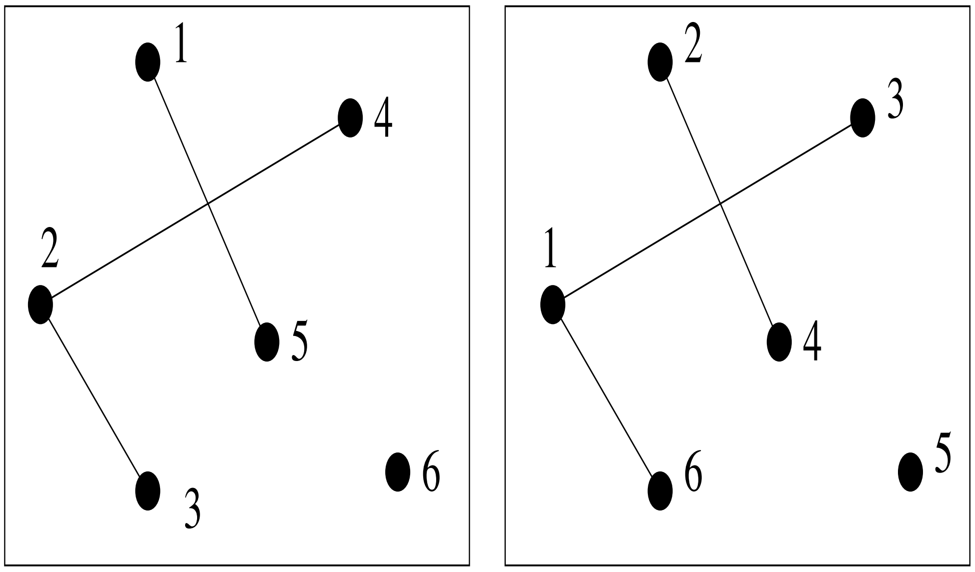

Figure 2: Two isomorphic graphs.

Protocol: Private-coin Graph Non-isomorphism

V: pick $i \in \{1, 2\}$ uniformly randomly. Randomly permute the vertices of G_i to get a new graph H. Send H to P.

P: identify which of G_1, G_2 was used to produce H. Let G_j be that graph. Send j to V.

V: accept if $i = j$; reject otherwise.

To see that definition 1 is satisfied by the above protocol, note that if $G_1 \not\equiv G_2$ then there exists a prover such that $\Pr[V \text{ accepts}] = 1$, because if the graphs are non-isomorphic, an all-powerful prover can certainly tell which one of the two is isomorphic to H. On the other hand, if $G_1 \equiv G_2$ the best any prover can do is to randomly guess, because a random permutation of G_1 looks exactly like a random permutation of G_2; that is $\Pr[V \text{accepts}] \leq 1/2$.

The above example depends crucially on the fact that P cannot see the random bits of V. If P knew those bits, P would know i and so could trivially always guess correctly. By comment (i), any problem with a private-coin interactive proof with a constant number of rounds has a public-coin proof. We now present such a protocol for Graph Non-isomorphism.

Example 2 (Public-coin protocol for Graph Non-isomorphism) To develop a public-coin protocol, we need to look at the problem in a different way. Consider the set $S = \{H : H \equiv G_1 \text{ or } H \equiv G_2\}$. The size of this set depends on whether G_1 is isomorphic to G_2. For a graph of n vertices, there are $n!$ possible ways to label the vertices, so we have

$$\begin{aligned} &\text{if } G_1 \not\equiv G_2 \text{ then } |S| = 2n! \\ &\text{if } G_1 \equiv G_2 \text{ then } |S| = n! \end{aligned}$$

We can amplify the gap between the two cases. Choose an integer m such that $n! \leq 2^m$. Let S' be the Cartesian product of S with itself a sufficient number of times so that the following hold:

$$\text{if } G_1 \not\equiv G_2 \text{ then } \left|S'\right| \geq 100 \cdot 2^m \tag{A}$$

$$\text{if } G_1 \equiv G_2 \text{ then } \left|S'\right| \leq \frac{1}{10} 2^m \tag{B}$$

We can now use the size of S' to determine if $G_1 \equiv G_2$. The question that remains is: how do we design an interactive protocol that can distinguish between cases (A) and (B) above?

Let $\mathcal{H}$ be a set of 2-universal hash functions from $\mathcal{U}$ to $\{0,1\}^m$, where $\mathcal{U}$ is a superset of S'. Given $h \in \mathcal{H}$, if S' is much bigger than $\{0,1\}^m$, as it is in case (A), then for every $y \in \{0,1\}^m$ it is very likely that there is some $x \in S'$ such that $h(x) = y$. Conversely, if the size of S' is much smaller than $\{0,1\}^m$, —if (B) holds— then for most $y \in \{0,1\}^m$ there is no $x \in S'$ such that $h(x) = y$. This is the basis of our protocol: V gives P an $h \in \mathcal{H}$ and $y \in \{0,1\}^m$, and if P can find an $x \in S'$ that h maps to y, the verifier guesses that the set is large, otherwise, we guess that the set is small. See figure 3. More formally, we have:

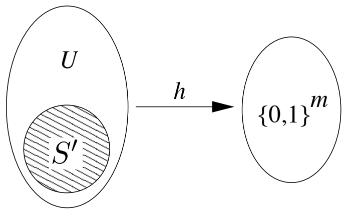

Figure 3: The bigger S' is, the more likely $h(S')$ will hit a given point in $\{0,1\}^m$.

Protocol: Goldwasser-Sipser Set Lowerbound

V: Randomly pick $h \in \mathcal{H}$, and $y \in_R \{0,1\}^m$. Send h, y to P.

P: Try to find an $x \in S'$ such that $h(x) = y$. Send such an x to V, or send a random element in $\mathcal{U}$ if no such x exists.

V: If $x \in S'$ and $h(x) = y$, accept; otherwise reject.

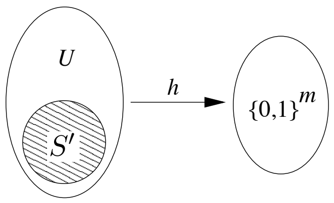

Figure 3: The bigger S' is, the more likely $h(S')$ will hit a given point in $\{0,1\}^m$.

Protocol: Goldwasser-Sipser Set Lowerbound

V: Randomly pick $h \in \mathcal{H}$, and $y \in_R \{0,1\}^m$. Send h, y to P.

P: Try to find an $x \in S'$ such that $h(x) = y$. Send such an x to V, or send a random element in $\mathcal{U}$ if no such x exists.

V: If $x \in S'$ and $h(x) = y$, accept; otherwise reject.

It remains to be shown that the above protocol fits Definition 1. Suppose (B) holds, then since the domain (S') is 1/10th the size of the range ($\{0,1\}^m$), we have $\Pr[V$ accepts if $G_1 \equiv G_2] \leq 1/10$. To find the probability of acceptance if case (A) holds we need a lemma that tells us that if S' is bigger than a certain size, any hash function we pick will likely map S' to a large fraction its range. The following lemma's proof is left as an exercise:

Lemma 1
Suppose $S \geq \mu 2^m$. *Then* $\Pr_{h \in \mathcal{H}}\left[|h(S)| \geq \left(1 - \frac{1}{\sqrt{\mu}}\right) 2^m\right] \geq 1 - \frac{1}{\sqrt{\mu}}$.

Here we have $\mu = 100$. Even if the hash function that we choose in the protocol is good, there is still a chance $1/\sqrt{\mu}$ that we will pick a $y \in \{0,1\}^m$ that is not mapped to by S. Hence,

$$\Pr[V \text{ accepts if } G_1 \equiv G_2] \geq 1 - \frac{1}{\sqrt{100}} - \frac{1}{\sqrt{100}} = \frac{8}{10}.$$

That Graph Non-isomorphism $\in$ **AM** follows from the definition.

We now turn to our main theorem.

Theorem 2 (LFKN, Shamir, 1990)
IP = **PSPACE**.

Proof: By comment (iv), we need only show that **PSPACE** $\subseteq$ **IP**[$poly(n)$]. To do so, we'll show that TQBF $\in$ **IP**[$poly(n)$]. This is sufficient because every $L \in$ **PSPACE** is polytime reducible to TQBF.

Again, we change the representation of the problem. For any Boolean formula of n variables $\phi(b_1, b_2, \ldots, b_n)$ there is a polynomial $P_\phi(x_1, x_2, \ldots, x_n)$ that is 1 if ϕ is true and 0 if ϕ is false. To see that this is true, consider the following correspondence between formulas and polynomials:

$$\begin{aligned} x \wedge y &\longleftrightarrow X \cdot Y \\ x \vee y &\longleftrightarrow 1-(1-X)(1-Y) \\ \neg x &\longleftrightarrow 1-X \end{aligned}$$

For example, $\phi = x \vee y \vee \neg z \longleftrightarrow 1-(1-X)(1-Y)Z$. If ϕ is a 3CNF formula with n variables and m clauses then we can write such a polynomial for each clause and multiply those polynomials to get a polynomial P_ϕ in n variables, with degree at most m in each variable. This conversion of ϕ to P_ϕ is called *arithmetization* and will be useful in our protocol.

Rather than tackle the job of finding a protocol for TQBF right away, we first present a protocol for $\#\mathbf{SAT}_L$, where

$$\#\mathbf{SAT}_L = \{\langle \phi, K\rangle : K \text{ is the number of sat. assignments of } \phi\}.$$

and ϕ is a 3CNF formula of n variables and m clauses.

Theorem 3
#SAT$_L$ $\in$ IP

Proof: Given ϕ, we construct, by arithmetization, P_ϕ. The number of satisfying assignments $\#\phi$ of ϕ is:

$$\#\phi = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\phi(b_1, \ldots, b_n) \tag{1}$$

There is a general protocol, *Sumcheck*, for verifying equations such as (1).

Sumcheck protocol. Given a degree d polynomial $g(x_1, \ldots, x_n)$ and an integer K, we present an interactive proof for the claim

$$K = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, \ldots, x_n). \tag{2}$$

V simply needs to be able to arbitrarily evaluate g. Define

$$h(x_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, b_2 \ldots, b_n).$$

If (2) is true, it must be the case that $h(0) + h(1) = K$.

To start, V randomly picks a prime p in the interval $[n^3, n^4]$ and instructs the prover to reduce all numbers modulo p in the remaining protocol. (For instance, if the prover wants to send a polynomial, it only needs to send the coefficients modulo p.) All computations described below are also modulo p. Consider the following protocol:

Protocol: Sumcheck protocol to check claim (2)

V: If $n = 1$ check that $g(1) + g(0) = K$. If so accept, otherwise reject. If $n \geq 2$, ask P to send $h(x_1)$ as defined above.

P: Send $h(x_1)$.

V: Let $s(x_1)$ be the polynomial received. Reject if $s(0) + s(1) \neq K$; otherwise pick a random a. Recursively use this protocol to check that

$$s(a) = \sum_{b \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2, \ldots, b_n).$$

Protocol: Sumcheck protocol to check claim (2)

V: If $n = 1$ check that $g(1) + g(0) = K$. If so accept, otherwise reject. If $n \geq 2$, ask P to send $h(x_1)$ as defined above.

P: Send $h(x_1)$.

V: Let $s(x_1)$ be the polynomial received. Reject if $s(0) + s(1) \neq K$; otherwise pick a random a. Recursively use this protocol to check that

$$s(a) = \sum_{b \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2, \ldots, b_n).$$

If Claim (2) is true, the prover that always returns the correct polynomial will always convince V. We prove by induction on n that if (2) is false, V rejects with high probability; we prove

$$\Pr[V \text{ rejects } \langle K, g\rangle] \geq \left(1 - \frac{d}{p}\right)^n. \tag{3}$$

With our choice of p, the right hand side is about $1 - dn/p$, which is very close to 1 since $p \geq n^3$.

We prove (3) by induction on n. The statement is true for $n = 1$ since V simply evaluates $g(0), g(1)$ and rejects with probability 1 if their sum is not K. Assume the hypothesis is true for degree d polynomials in $n - 1$ variables.

In the first round, the prover P is supposed to return the polynomial h. Suppose that P returns $s \neq h$. If $s(0) + s(1) \neq K$, then V rejects with probability 1. So assume that $s(0) + s(1) = K$. In this case, V picks a random a. If s and h are two different degree d polynomials, then there are at most d values of x_1 such that $s(x_1) = h(x_1)$. Thus,

$$\Pr_a[s(a) \neq h(a)] \geq 1 - \frac{d}{p}. \tag{4}$$

If $s(a) \neq h(a)$ then the prover is left with an incorrect claim to prove in the recursive step. By the induction hypothesis, with probability $\geq \left(1 - \frac{d}{p}\right)^{n-1}$, P cannot prove this false claim. Thus we have

$$\Pr[V \text{ rejects}] \geq \left(1 - \frac{d}{p}\right) \cdot \left(1 - \frac{d}{p}\right)^{n-1} = \left(1 - \frac{d}{p}\right)^n \tag{5}$$

This finishes the induction.

We still have to justify why it is OK to perform all operations modulo p. The fear, of course, is that equation (2) might be false over the integers, but true over p, which happens if p divides the difference of the two sides of (2). The following lemma implies that the chance that this happens for a random choice of p is small, since an n-bit integer (which is what K is) has at most n prime divisors.

Lemma 4 (Fingerprinting)
Suppose x, y, $x \neq y$ are n-bit integers. Then there are at most n primes that divide $|x - y|$.

An interactive proof for $\#\mathbf{SAT}_L$ is obtained by letting $g = P_\phi$. $\square$

We use a very similar idea to obtain a protocol for TQBF. Given a true, fully qualified Boolean formula $\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n \phi(x_1, \ldots, x_n)$, we use arithmetization to construct the polynomial P_ϕ. We have that $\phi \in$ TQBF if and only if

$$0 < \sum_{b_1 \in \{0,1\}} \prod_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} \cdots \prod_{b_n \in \{0,1\}} P_\phi(b_1, \ldots, b_n)$$

A first thought is that we could use the same protocol as in the $\#\mathbf{SAT}_L$ case, except check that $s(0) \cdot s(1) = K$ when you have a $\prod$. But, alas, multiplication, unlike addition, increases the degree of the polynomial — after n steps when V must evaluate g directly, the degree could be 2^n. There could be exponentially many coefficients. The solution is to observe that we are only interested in $\{0,1\}$ values. You can always approximate a polynomial with a multi-linear function if you only evaluate it at $\{0,1\}^n$. Let Rx_i be a linearization operator defined as

$$Rx_1[p(x_1, \ldots, x_n)] = (1 - x_1)p(0, x_2, \ldots, x_n) + (x_1)p(1, x_2, \ldots, x_n). \tag{6}$$

Now, instead of working with $\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n \phi(x_1, \ldots, x_n)$, work with

$$\exists x_1 Rx_1 \forall x_2 Rx_1 Rx_2 \exists x_3 Rx_1 Rx_2 R_x 3 \cdots \phi(x_1, \ldots, x_n) \tag{7}$$

The size of the expression is $1 + 2 + 3 + \cdots + n \approx n^2$. The protocol for $\#\mathbf{SAT}_L$ can be used suitably modified, where in rounds involving the linearization operator, the verifier uses (6). The use of the linearization operator ensures that the polynomial which the prover needs to send at every round is linear, and hence the blowup in degrees is avoided. $\square$