1 Characterizing IP

Recall that in an interactive proof for a language \( L \) we have a computationally unbounded prover \( P \) and a verifier \( V \) with the properties:

- Efficiency: \( V \) runs in time \( \text{poly}(|x|) \)
- Completeness: \( x \in L \rightarrow \Pr[V \text{ accepts in } (P,V)(x)] \geq 2/3 \)
- Soundness: \( x \notin L \rightarrow \forall P^*, \Pr[V \text{ accepts in } (P^*,V)(x)] \leq 1/3 \)

Last time we showed that \( \text{P}^\#P \subseteq \text{IP} \). In fact:

**Theorem 1** \( \text{IP} = \text{PSPACE} \)

**Proof:** (sketch)

\( \subseteq \): Homework, PS5

\( \supseteq \): The proof is similar to \( \text{P}^\#P \subseteq \text{IP} \), using similar arithmetic techniques to transform the problem into one of polynomials over finite fields, except we use TQBF instead of \#SAT. In contrast to the summation in \#SAT, handling quantifiers in TQBF causes the degree of polynomial to increase exponentially, so a clever “degree reduction” trick is needed to keep it small.

### 1.1 Nice Properties of \#SAT and TQBF Proof Systems

1. The prover for both can be implemented in \( \text{P}^L \) - there is no need for anything stronger. This does not appear to be true for all languages with interactive proofs.

2. Perfect completeness - In both systems if \( x \in L \), the verifier accepts always. This implies that every language in \( \text{IP} \) has a perfectly complete interactive proof. (Since \( \text{IP} \subseteq \text{PSPACE} \), every language \( L \in \text{IP} \) reduces to TQBF, so we can obtain a new, perfectly complete interactive proof that \( x \in L \) by reducing \( x \) to an instance of TQBF and applying the protocol for TQBF.)

3. Public coins - The verifier in either case needs no hidden randomness. This implies that every language in \( \text{IP} \) has a public-coin protocol, including graph nonisomorphism. (Although public coins may come at the cost of efficiency). Note that the prover still cannot see future coins of the verifier.
2 Consequences for Program Checking

Definition 2
A program checker (a.ka. instance checker) for \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) is a PPT \( M \) such that for all inputs \( x \):

1. Completeness: \( \Pr[M^f(x) \text{ accepts}] \geq 2/3 \) (or \( = 1 \) for perfect completeness)
2. \( \forall g \text{ such that } g(x) \neq f(x), \Pr[M^g(x) \text{ accepts}] \leq 1/3 \)

Idea: someone claims that a program \( g \) computes the the function \( f \). We want to use \( g \) to compute \( f \) on an input \( x \), but we are concerned that \( g \) may be incorrect (either due to bugs or to being malware). By running \( M^g(x) \) we can be confident that we won’t accept an incorrect value \( g(x) \).

Proposition 3
If \( L \) and \( \overline{L} \) have interactive proof systems where the prover can be implemented in \( P^L \) (or equivalently \( P^{\overline{L}} \)), then \( L \) has a program checker.

Proof:
Given an oracle \( L^* \) to be checked, our program checker is \( M^{L^*}(x) : \)

- Query \( L^*(x) \) and let \( y \in \{0,1\} \) be the result.
- If \( y = 1 \) simulate the IP for \( L \) to verify that \( x \in L \).
- If \( y = 0 \) simulate the IP for \( \overline{L} \) to verify that \( x \notin L \)
- Accept/reject accordingly

As a result, Graph Isomorphism, \#SAT, TQBF all have program checkers because of this. Note that the above does not show that all of \( IP = PSPACE \) has program checkers, because we require that the prover be implementable with oracle access to \( L \), rather than to a \( PSPACE \)-complete problem. In fact, it is an open problem whether SAT has a program checker, and the best known interactive proof for \( coNP \) still requires a \( \#P \) oracle!

3 Arthur–Merlin Games

Definition 4
A public-coin interactive proof is an interactive proof \( (P,V) \) where each message from \( V \) consists of uniformly random coins and at the end \( V \) accepts by a deterministic poly-time function of \( x \) and the transcript of communications between \( P \) and \( V \).
This is also sometimes known as an Arthur-Merlin protocol, where we imagine Merlin, an all-powerful prover, trying to convince Arthur, the limited verifier, of something.
Definition 5
For a function $k : \mathbb{N} \rightarrow \mathbb{N}$…

$\text{IP}[k(n)] = \{ L : L \text{ has interactive proofs with } \leq k(n) \text{ messages} \}$

$\text{IP} = \bigcup_n \text{IP}[n^c]$

$\text{AM}[k(n)] = \{ L : L \text{ has public-coin interactive proofs with } \leq k(n) \text{ messages and Arthur speaks first} \}$

$\text{MA}[k(n)] = \{ L : L \text{ has public-coin interactive proofs with } \leq k(n) \text{ messages and Merlin speaks first} \}$

$\text{AM} = \text{AM}[2]$

$\text{MA} = \text{MA}[2]$

We present the following facts:

• $\text{IP}[\text{poly}(n)] = \text{AM}[\text{poly}(n)]$ because only public-coins were needed in $\text{IP} = \text{PSPACE}$.

• $\forall k(n) \geq 2$, $\text{IP}[k(n)] = \text{AM}[k(n)]$. In particular, GNI $\in \text{AM}[2]$. Loosely, “public coins = private coins”. (We’ll prove the case $k(n) = 2$ next time.)

• $\forall k(n) \geq 2$, $\text{MA}[k(n)] \subseteq \text{AM}[k(n)]$. (PS 5)

• $\forall k(n) \geq 2$, $\text{AM}[k(n)]$ with perfect completeness. (Possibly to be done in section.)

• $\forall k(n) \geq 2$, $\text{MA}[k(n)]$ with perfect completeness. (Possibly to be done in section.)

• $\forall k(n) \geq 2$, $\forall c$ constant, $\text{AM}[ck(n)] = \text{AM}[k(n)]$. In particular, $\text{AM}[c] = \text{AM}[2]$. (PS 5)

3.1 Relationships of AM and MA to NP

In $\text{MA}$, we have $M$ sending $m$, then $A$ tossing coins $r$, and then a deterministic verifier $A(x, m, r)$. By completeness and soundness:

$x \in L \rightarrow \Pr_r[\exists m, A(x, r, m) = 1] \geq 2/3$

$x \notin L \rightarrow \Pr_r[\exists m, A(x, r, m) = 1] \leq 1/3$

This is exactly $\text{NP}$ except with a $\text{BPP}$ verifier, instead of a $\text{P}$ verifier!

In $\text{AM}$, we have $A$ sending coins $r$, then $M$ sending $M$, and then a deterministic verifier $A(x, m, r)$. By completeness and soundness:

$x \in L \rightarrow \Pr_r[\exists m_r, A(x, r, m) = 1] \geq 2/3$

$x \notin L \rightarrow \Pr_r[\exists m_r, A(x, r, m) = 1] \leq 1/3$

This is a randomized version of $\text{NP}$, where we have some randomness at the beginning, and then afterwards, check an $\text{NP}$-like condition that depends on the randomness.

On PS5 you will show $\text{MA} \subseteq \text{AM}$, and thus we have inclusions:
4 Approximate Counting $\in \text{AM}$

**Theorem 6**
For every $f \in \#P$ and every constant $\alpha > 1$ (or even $\alpha = 1 + 1/\text{poly}(n)$), we have $\text{GAP}_\alpha f \in \text{prAM}$, where $\text{GAP}_\alpha f$ is the promise problem:

yes: $\{(x,t) : f(x) \geq t\}$
no: $\{(x,t) : f(x) < t/\alpha\}$

**Corollary 7**
Approximate counting and almost-uniform sampling are both in $\text{BPP}^{\text{NP}}$.

**Proof:**
We show the theorem true for $\alpha = 4$. Next time we’ll show how to deduce it for $\alpha = 1 + 1/\text{poly}(n)$.

$f \in \#P$, so by definition $f(x) = |S(x)|$ for some $\text{NP}$ search problem $S$. We give a $\text{prAM}$ protocol using hashing. Given $(x, t)$:

1. Arthur chooses $m \in \mathbb{N}$ such that $2^{m-1} > t \geq 2^{m-2}$, picks pairwise-independent hash $h : \{0,1\}^{\text{poly}(n)} \rightarrow \{0,1\}^m$ and sends $h$ to Merlin.

2. Merlin finds $y \in S(x)$ such that $h(y) = 0^m$ and sends $y$.

3. Arthur accepts if $h(y) = 0^m$ and $y \in S(x)$.

Completeness: If $|S(x)| \geq t \geq 2^{m-2}$ then by the Valiant-Vazirani analysis, with probability $\geq 1/8$ there exists some element in $S(x)$ mapping to 0.

Soundness: If $|S(x)| < t/\alpha < 2^{m-1}/\alpha = 2^{m-3}$ then the probability that there exists some element in $S(x)$ mapping to 0 is, by union bound, $\leq \sum_{y \in S(x)} \Pr_{h}[h(y) = 0^m] = |S(x)|/2^m \leq 2^{m-3}/2^m \leq 1/16$. 

4
Since we have a finite gap $1/8$ to $1/16$, we can amplify as desired, giving us an prAM protocol.