Characterizing the Power of IP

Xin Li

lixints@cs.jhu.edu
Topics

• IP for coNP.

• IP vs PSPACE.
Graph Isomorphism

• Question: can GI be NP-complete?

• Theorem: If GI is NP-complete, then $PH=\Sigma_2^p$.

• The proof only uses the following two facts: (1) GNI $\in$ AM. (2) If GI is NP-complete, then GNI is coNP-complete.

• So we have the following theorem: If $coNP \subseteq AM$, then $PH=\Sigma_2^p$. 
Interactive Proof for ¬ 3SAT

• We know ¬ 3SAT is coNP-complete.

• From previous theorem ¬ 3SAT is unlikely to have constant round interactive proof.

• We give a public coin interactive proof for ¬ 3SAT with n rounds.

• n: # of variables in the given 3CNF formula Φ.
Interactive Proof for \( \neg \text{3SAT} \)

- How can the prover convince the verifier that \( \Phi \) has no satisfying assignment?

- We show how to prove something more general.

- Let \( \#\text{SAT}=\{< \Phi, k >, \Phi \text{ is a 3CNF formula that has exactly } k \text{ satisfying assignments}\} \).

- Show an interactive proof for \( \#\text{SAT} (\neg \text{3SAT} \text{ is a special case}) \).
Interactive Proof for \( \neg \text{3SAT} \)

- \#SAT=\{\langle \Phi, k \rangle, \Phi \text{ is a 3CNF formula that has exactly } k \text{ satisfying assignments}\}.
- The set lower bound protocol can be used to certify \( k \) approximately with 2 rounds.
- We design an \( n \)-round protocol that certifies \( k \) exactly.
- Idea: arithmetization.
Interactive Proof for $\neg$ 3SAT

• Represent Boolean formulas by polynomials. True: 1, False: 0

• Example: $x_i \land x_j \Rightarrow x_i x_j$; $x_i \lor x_j \Rightarrow 1-(1-x_i)(1-x_j)$.

• Given any 3CNF formula $\Phi (x_1, \ldots, x_n)$ with $m$ clauses and $n$ variables, we introduce $n$ field variables $x_1, \ldots, x_n$.

• For any clause of size 3 we write an equivalent degree 3 polynomial.
Interactive Proof for \( \neg 3\text{SAT} \)

• For any clause of size 3 we write an equivalent degree 3 polynomial.

• Denote the polynomial for the j’th clause by \( P_j (x_1, \ldots, x_n) \).

• For every 0, 1 assignment to \( x_1, \ldots, x_n \), we have \( P_j (x_1, \ldots, x_n) = 1 \) iff the assignment satisfies the j’th clause.

• Multiplying these polynomials, we get a multivariate polynomial \( P_\Phi (x_1, \ldots, x_n) = \prod_j P_j (x_1, \ldots, x_n) \).
Interactive Proof for $\neg$ 3SAT

- Multivariate polynomial $P_\Phi(x_1, \ldots, x_n) = \prod_j P_j(x_1, \ldots, x_n)$.

- $P_\Phi(x_1, \ldots, x_n)$ is 1 iff the assignment satisfies $\Phi$.

- The degree of $P_\Phi$ is at most $3m$.

- Always represent $P_\Phi$ as the product degree 3 polynomials, so $P_\Phi$ has size $O(m)$. 
Interactive Proof for $\neg \text{3SAT}$

- **Arithmetization**: we have converted a Boolean formula $\Phi$ into a polynomial $P_\Phi$.

- Now, we can substitute arbitrary values from the field $F$ to $P_\Phi$, instead of just 0, 1.

- This gives the verifier additional power.

- We can now give the interactive proof for #SAT.
Interactive Proof for ¬ 3SAT

• **Given input** \(< \Phi , k >\), both P and V construct the polynomial \(P_\Phi\).

• **The # of satisfying assignments** \(#\Phi\) of \(\Phi\) can be computed as
  \[ #\Phi = \sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} P_\Phi(b_1, b_2, \ldots, b_n) \]

• **The prover’s claim is that** \(#\Phi\) **is exactly equal to** \(k\).

• **From now on, forget about the Boolean formula and focus on this claim.**
Interactive Proof for $\neg$ 3SAT

• To start, $P$ sends $V$ a prime number $p$ in the range $(2^n, 2^{2n}]$.

• $V$ can check $p$ is indeed a prime number in polynomial time.

• Now we think of all variables $x_1, \ldots, x_n$ to be in the field $F_p$. That is, \{0, 1, \ldots, p-1\} and all the operations are mod $p$.

• We give a general sumcheck protocol for verifying equations in $F_p$. 
Sumcheck protocol

• Given a degree d polynomial \( g(x_1, \ldots, x_n) \), an integer \( k \) and a prime \( p \).

• Show an interactive proof for the claim \( k=\sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(b_1, b_2, \ldots, b_n) \mod p \).

• The only property \( V \) needs is that \( g \) has a poly(n) size representation, and thus \( V \) can compute \( g \) in poly-time.

• Note that this also solves \#SAT since \( \#\Phi \leq 2^n \) and \( p > 2^n \).
Sumcheck protocol

• An interactive proof for the claim $k = \sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(b_1, b_2, \ldots, b_n) \mod p$.

• Observation: if fix $x_2 = b_2$, $x_3 = b_3$, ..., $x_n = b_n$, then $g(x_1, b_2, \ldots, b_n)$ is a degree $d$ univariate polynomial in $x_1$.

• Thus, $h(x_1) = \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(x_1, b_2, \ldots, b_n)$ is also a degree $d$ univariate polynomial in $x_1$.

• If the claim is true, then we must have $h(0) + h(1) = k$. 
Sumcheck protocol

- An interactive proof for the claim $k = \sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(b_1, b_2, \ldots, b_n) \mod p$.

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

- V: If $n=1$, directly check that $g(1)+g(0)=k$. If so accept; otherwise reject. If $n \geq 2$, ask P to send the description of $h(x_1)$ (d+1 coefficients).

- P: send some polynomial $s_1(x_1)$ that is supposed to be $h(x_1)$. 
Sumcheck protocol

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

- $V$: If $n=1$, directly check that $g(1)+g(0)=k$. If so accept; otherwise reject. If $n \geq 2$, ask $P$ to send the description of $h(x_1)$ ($d+1$ coefficients).

- $P$: send some polynomial $s_1(x_1)$ that is supposed to be $h(x_1)$.

- $V$: Reject if $s_1(0)+s_1(1) \neq k$. Otherwise pick a random $a$ in $F_p$ and recursively use the same protocol to check $s_1(a) = \sum_{b_2 \in \{0, 1\} \ldots \sum_{b_n \in \{0, 1\}} g(a, b_2, \ldots, b_n)$. 
**Sumcheck protocol**

- **Main idea:** V cannot compute the sum by himself, so ask P for help by requiring $h(x_1)$.

- However, P may cheat by sending some $s_1(x_1) \neq h(x_1)$.

- So V pick a random $a$ in $F_p$ and check $s_1(a) = h(a) = \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(a, b_2, \ldots, b_n)$.

- If P cheats, then each time there is some probability of catching P.
Sumcheck protocol

- **Claim:** $k = \sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} g(b_1, b_2, \ldots, b_n) \mod p$ (*).

- If (*) holds, certainly $P$ can convince $V$ to accept by sending the correct polynomials.

- **Claim:** if (*) is false, then $V$ rejects with probability $\geq (1-d/p)^n \geq 1- dn/p$.

- **Proof by induction.**
Sumcheck protocol

• Claim: if (*) is false, then $V$ rejects with probability $\geq (1-d/p)^n \geq 1- \frac{dn}{p}$.

• Base case $n=1$: $V$ rejects with probability 1.

• Assume the claim holds for degree $d$ polynomials with $n-1$ variables.

• Consider the round where $P$ sends $s_1(x_1)$. If $s_1(x_1) = h(x_1)$ and (*) is false, $V$ will reject immediately.
Sumcheck protocol

• Claim: if (*) is false, then V rejects with probability $\geq (1-d/p)^n \geq 1- \frac{dn}{p}$.

• If $P$ sends some $s_1(x_1) \neq h(x_1)$, then $s_1(x_1) - h(x_1) \neq 0$ and is a degree $d$ polynomial. Thus it has at most $d$ roots. That is, at most $d$ choices of $a$ s.t. $s_1(a) = h(a)$.

• Thus, if $V$ chooses a randomly from $F_p$, then $Pr[s_1(a) \neq h(a)] \geq 1- \frac{d}{p}$.

• In this case $P$ is left with an incorrect claim to prove. By inductive hypothesis $V$ rejects with probability $(1-d/p)^{n-1}$. Thus overall $V$ rejects with probability $\geq (1-d/p)^n$. 
Sumcheck protocol

- Recall, previously we designed a sumcheck protocol for \# SAT:

\[
\# \Phi = \sum_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \sum_{b_n \in \{0, 1\}} P_\Phi(b_1, b_2, \ldots, b_n) = k.
\]

- This shows that \# SAT and \neg 3SAT \in IP.

- More generally, coNP \subseteq IP.
Generalize to TQBF

• Show a poly round interactive proof for TQBF.

• This implies that PASPACE $\subseteq$ IP.

• Since we already showed that IP $\subseteq$ PASPACE, we have IP = PASPACE.

• Idea: generalize the sumcheck protocol.
Protocol for TQBF

- TQBF: given a QBF $\psi = Q_1 x_1 \, Q_2 \, x_2 \ldots \, Q_n \, x_n \, \Phi(x_1, x_2, \ldots x_n)$ with $n$ variables and size $m$, decide if $\psi$ is True.

- Example: $\psi = \forall \, x_1 \, \exists \, x_2 \ldots \, \exists \, x_n \, \Phi(x_1, x_2, \ldots x_n)$.

- Use arithmetization to construct the polynomial $P_\Phi$ such that $\psi \in TQBF$ iff 

- $\Pi_{b_1 \in \{0, 1\}} \sum_{b_2 \in \{0, 1\}} \ldots \, \sum_{b_n \in \{0, 1\}} P_\Phi(b_1, b_2, \ldots, b_n) \neq 0$. 
Protocol for TQBF

• $\prod_{b_1 \in \{0, 1\}} \Sigma_{b_2 \in \{0, 1\}} \ldots \Sigma_{b_n \in \{0, 1\}} P_\Phi(b_1, b_2, \ldots, b_n) \neq 0$ (or $= k$).

• First thought: use the same protocol as for #SAT, except for $x_1$ (or any variable with $\forall$), need to check if $s_1(0) \cdot s_1(1) = k$.

• Problem: the degree of the polynomial can be as large as $2^n$ (due to $\Pi$).

• The polynomial $V$ ask $P$ to send can have $2^n$ coefficients.
Protocol for TQBF

- $\Pi_{b_1 \in \{0, 1\}} \Sigma_{b_2 \in \{0, 1\}} \ldots \Sigma_{b_n \in \{0, 1\}} P_{\Phi}(b_1, b_2, \ldots, b_n) \neq 0$ (or $=k$) (*)

- Need to find a way to reduce the degree.

- Key observation: (*) only uses 0,1 values, for $x=0/1$, $x^t=x$ for all $t$.

- Thus, can convert any polynomial $p(x_1, \ldots, x_n)$ into a multilinear polynomial $q(x_1, \ldots, x_n)$ that agrees with $p$ on all $x_1, \ldots, x_n \in \{0,1\}$. 
Protocol for TQBF

- Linearization operator: \( \forall \) polynomial \( p(.) \), let \( L_i (p) \) be the polynomial defined as follows

\[
L_i (p)(x_1, \ldots, x_n) = x_i p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + (1-x_i) p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).
\]

- \( L_i (p) \) is linear in \( x_i \) and agrees with \( p(.) \) on whenever \( x_i \in \{0,1\} \).

- \( L_1 ( L_2 ( \ldots (L_n (p) \ldots ))) \) is a multilinear polynomial that agrees with \( p \) on all \( x_1, \ldots, x_n \in \{0,1\} \).
Protocol for TQBF

- We now view \( \forall \) and \( \exists \) as operators on polynomials as well:

- \( \forall x_i \, p(x_1, \ldots, x_n) = p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \times p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n). \)

- \( \exists x_i \, p(x_1, \ldots, x_n) = p(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + p(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n). \)

- We can now rephrase claim (*).
Protocol for TQBF

- \( \Pi_{b_1 \in \{0, 1\}} \Sigma_{b_2 \in \{0, 1\}} \ldots \Sigma_{b_n \in \{0, 1\}} P_\Phi(b_1, b_2, \ldots, b_n) = k \) (*)

- Rephrase: if we apply a sequence of operators \( \forall x_1 \exists x_2 \ldots \exists x_n \) on \( P_\Phi(x_1, x_2, \ldots, x_n) \) then we get a (non-zero) number \( k \).

- We can insert any arbitrary sequence of linearization operators into this sequence to make sure all intermediate polynomials have low degree.

- Use \( Q= \forall x_1 \ L_1 \exists x_2 \ldots L_1 \ L_2 \ldots L_{n-1} \exists x_n \ L_1 \ L_2 \ldots L_n \ P_\Phi(x_1, x_2, \ldots, x_n) \). Size=\( O(m+n^2) \)
Protocol for TQBF

- $Q = \forall x_1 L_1 \exists x_2 \ldots L_1 L_2 \ldots L_{n-1} \exists x_n L_1 L_2 \ldots L_n g(x_1, x_2, \ldots, x_n) = k \ (*)$

- $V$: If $n=1$, directly check the equality. If it holds accept; otherwise reject

- $V$: If $n \geq 2$, use recursion.

- Notice that whenever we have a $\forall x_i$ or $\exists x_i$, all variables after $x_i$ will disappear after the operator.
Protocol for TQBF

\[ Q = \forall x_1 L_1 \exists x_2 \ldots L_2 \ldots \exists x_{n-1} L_1 L_2 \ldots L_n g(x_1, x_2, \ldots, x_n) \]

- **Variables fixed in previous rounds**
- **Look at this operator in this round**
- **The rest is a polynomial in**
  \[ x_1, x_2, \ldots, x_t \text{ for some } t. \]
Protocol for TQBF

• Suppose at some level of recursion, we have $U(x_1, x_2, \ldots, x_l) = O(g(x_1, x_2, \ldots, x_l))$, where $O$ is $\forall x_t, \exists x_t$ or $L_i$.

• $P$ tries to convince $V$ that $U(a_1, a_2, \ldots, a_l) = c$. Let $d \leq \text{poly}(n)$ be the degree of each $x_i$ in $U$.

• $\exists x_t$: in this case $l = t - 1$. $P$ sends a degree $d$ polynomial $s(x_t)$ supposed to be $g(a_1, a_2, \ldots, a_t, x_t)$.

• $V$ checks if $s(0) + s(1) = c$. If not reject; otherwise pick a random $a$ in $F_p$ and ask $P$ to prove $s(a) = g(a_1, a_2, \ldots, a_{t-1}, a)$.
Protocol for TQBF

• Suppose at some level of recursion, we have $U(x_1, x_2, \ldots, x_l) = O(g(x_1, x_2, \ldots, x_l))$, where $O$ is $\forall x_t, \exists x_t$ or $L_i$.

• $P$ tries to convince $V$ that $U(a_1, a_2, \ldots, a_l) = c$. Let $d \leq \text{poly}(n)$ be the degree of each $x_i$ in $U$.

• $\forall x_t$: in this case $l = t - 1$. $P$ sends a degree $d$ polynomial $s(x_t)$ supposed to be $g(a_1, a_2, \ldots, a_{t-1}, x_t)$.

• $V$ checks if $s(0) \cdot s(1) = c$. If not reject; otherwise pick a random $a$ in $F_p$ and ask $P$ to prove $s(a) = g(a_1, a_2, \ldots, a_{t-1}, a)$. 
Protocol for TQBF

• Suppose at some level of recursion, we have $U(x_1, x_2, \ldots, x_l) = O(g(x_1, x_2, \ldots, x_t))$, where $O$ is $\forall x_t \forall x_t \exists x_t$ or $L_i$.

• $P$ tries to convince $V$ that $U(a_1, a_2, \ldots, a_l) = c$. Let $d \leq \text{poly}(n)$ be the degree of each $x_i$ in $U$.

• $L_i$ : in this case $l=t$. $P$ sends a degree $d$ polynomial $s(x_i)$ supposed to be $g(a_1, a_2, \ldots, x_i, \ldots, a_l)$.

• $V$ checks if $a_i s(1) + (1-a_i) s(0) = c$. If not reject; otherwise pick a random $a$ in $F_p$ and ask $P$ to prove $s(a) = g(a_1, a_2, \ldots, a, \ldots, a_l)$. 
Protocol for TQBF

• The proof follows the same argument of induction as in #SAT.

• Completeness=1

• Soundness: each time P cheats, V can catch that with probability $\geq (1-d/p)$. So soundness $\leq 1-(1-d/p)^{n^2} \leq dn^2/p$.

• Notice that in the recursion, the assignments to $x_1, x_2, \ldots, x_n$ keep changing.
Multi-Prover Interactive Proof

- Previously, there is one prover and one verifier.

- Can generalize to more than one prover.

- Provers may communicate before the protocol to fix a strategy, but cannot communicate during the protocol (cannot collaborate).

- Similar to interrogating two suspects in separate rooms.
Multi-Prover Interactive Proof

- This leads to the class MIP.

- Turns out for MIP, two provers is equivalent to poly(n) provers.

- MIP=\text{NEXP}

- V can ensure the answer of a prover to a question q is a function only of q, and does not depend on previous ones, by asking another prover of the same question, and accept only if both answers agree.