Complexity Classes VII

Interactive Proofs

Eric Rachlin
Verifiers and Provers

- **NP**: Languages that can be recognized in PTIME with access to a certificate or proof.
- **PCP**: Languages that can be recognized in PTIME with limited access to a proof. Here the verifier uses random bits.
- **IP**: Languages that can be recognized in PTIME with access to "Prover", a machine that answers a series of questions.
- **MIP**: Like IP, only with multiple provers.
  - Historically, IP and MIP came before PCP
What good is a prover?

- Intuition: To prove a statement to you, I could write down a lengthy proof. Rather than verify a long proof, it’s easier to adaptively ask me questions.

- Example: Graph Isomorphism:
  - To show $G_1$ and $G_2$ are not isomorphic, a verifier can repeatedly pick one at random, randomly permute it, and ask the prover if the new graph came from $G_1$ or $G_2$.
  - If the prover is ``smart enough'', and the graphs really aren’t isomorphic, it will always answer correctly right.
  - If the two graphs are isomorphic, the prover will be unable to answer correctly more than half the time.
Modeling k-rounds of Deterministic Interaction

- Let \( x \) be a binary string, \( L \) be a language, and \( V \) and \( P \) be functions \( \{0, 1\}^n \rightarrow \{0,1\}^m \), where \( n \) and \( m \) are polynomial in \( |x| \).

- Consider the following k step interaction, \( V \) (but not necessarily \( P \)) is PTIME, and \( y \) indicates if \( x \) is in \( L \):
  - \( q_1 = V(x) \)
  - \( a_1 = P(x, q_1) \)
  - \ldots
  - \( q_k = V(x, q_1, a_1, \ldots, q_{k-1}, a_{k-1}) \)
  - \( a_k = P(x, q_1, a_1, \ldots, q_{k-1}, a_{k-1}, q_k) \)
  - \( y = V(x, q_1, a_1, \ldots, q_{k-1}, a_{k-1}, q_k) \)
IP[k] without randomness

- What languages can be recognized by k steps of deterministic interaction?
  - Since the verifier is deterministic, P, given x, could simply write down all k answers in advance.
  - The verifier could then verify these are the answers the verifier would have asked.
- What if we allow k to be polynomial in |x|?
  - If k is polynomial in |x|, the answers are polynomial in x, and the languages recognized are those in NP.
The Real Deal: IP

- Let $\text{IP}_{c,s}[k]$ be the languages recognized by $k$ rounds of interaction, where $V$ is now a random function.
  - $V$ always gets a random sequence $r$ as input, where $|r|$ is polynomial in $|x|$.
  - Now "recognized" is given in terms of completeness, $c$, and soundness, $s$.
- Let $\text{IP}$ be the languages recognized by $k(x)$ rounds of interaction, where $k(x)$ is any polynomial, $c = 2/3$ and $s = 1/3$. 

More on the Definition of IP

- It turns out IP is the same if $c = 1$
  - If $s = 0$, however, IP becomes NP.
- The power of P is unbounded, but in fact P can be restricted to functions in PSACE.
- We gave V access to a random string, why not P?
  - If a random verifier works, a deterministic verifier could just pick the answer most likely to succeed.
- Is it necessary that V’s random bits be kept secret from P?
  - $AM[k]$ denotes $IP[k]$ when random bits are public. It turns out $IP[k] \subseteq AM[k + 2]$, but this is not obvious.
IP ⊆ PSPACE

- Consider $k$ rounds of interaction:
  - $q_1 = V(x, r_1)$
  - $\ldots$
  - $a_k = P(x, q_1, a_1, \ldots, q_{k-1}, a_{k-1}, q_k)$
  - $y = V(x, q_1, a_1, \ldots, q_{k-1}, a_{k-1}, q_k, r_k)$

- $P$ is a PTIME function, in the absence of random bits, the probability of its output can be computed in PSPACE (just try every random sequence)

- Given $(x, q_1, a_1, \ldots, q_k)$, $a_k$ is PSPACE computable:
  - Try each $a_k$, pick the one for which $\text{PROB}[y = 1]$ is highest.

- Induction: since $a_k$ is PSPACE computable, so is $a_{k-1}$:
  - Again, try each $a_{k-1}$, find $a_k$, maximize $\text{PROB}[y = 1]$. 
PSPACE Completeness

- It turns out IP = PSPACE! (Shamir, 1990)
- To show this, show a PSPACE complete language is in IP.
- TQBF: x is in TQBF if it represents a true "Totally Quantified Boolean Formula":
  - A boolean formula, F(x₁, …xₙ), is a boolean expression comprised of AND’s, ORs and NOTs.
  - In TQBF, each variable appears in either an existential or universal quantifier to the left of F(x₁, …xₙ).
  - TQBF becomes SAT if quantifiers must be existential.
- TQBF is PSPACE Complete.
TQBF is PSPACE-Complete

- To show SAT is NP complete, the computation (or tableau) of a nondeterministic turning is represented as an existentially quantified boolean formula.
- To show TQBF is PSPACE complete, we represent a PSPACE computation using a quantified formula.
- Given a machine M, let $F_M(x,y,t)$ denote the boolean function `M goes from state x to state y in t steps".
- Notice that $F_M(x,y,t) = \exists w F_M(x,w,t/2) \land F_M(w,y,t/2)$.
  \[ = \exists w \forall x',y' \ ( (x',y' = x,w) \land (x',y' = w,y) ) \rightarrow F_M(x',y',t/2) \]
- The formula $F_M(x,y,1)$ can be implemented directly when a polynomial number of variables represent x and y.
- If M uses a polynomial amount of space, $F_M(x,y,t)$ can be reduced to a polynomial sized instance of TQBF in PTIME.
Formulas As Polynomials

- A boolean expression can be represented as an arithmetic expression as follows:
  - $x \land y = (xy)$
  - $x \lor y = (x + y)$
  - $\neg x = (1 - x)$

- If variables are binary, we can also let
  - $\exists x f(x, y)$ denote $\sum_x f(x, y) - \prod_x f(x, y) = 1$
  - $\forall x f(x, y)$ denote $\prod_x f(x, y) = 1$

- Also, if variables are binary, $x^k = x$ for $k \geq 1$
  - The polynomial representing any boolean formula, or partially quantified boolean formula is multilinear.
An IP for TQBF

- We check if the arithmetic expression representing a TQBF with $n$ variables evaluates to $K = 0$ or $K = 1$.
  - Arithmetic is mod some prime, $p$, provided by the prover.
- If $n = 1$
  - If universally quantified, check $K' = f(0)f(1)$
  - If existentially quantified, check $K' = f(0) + f(1) - f(0)f(1)$
- If $n > 1$
  - Let $z$ be the variable quantified by the left most quantifier.
  - Ask the prover for a linear function, $h(z) = f(x_1, \ldots, z, \ldots, x_n)$
    - If $z$ was universally quantified, check $K' = h(0)h(1)$
    - If existentially quantified, check $K' = h(0) + h(1) - h(0)h(1)$
  - Recurse: randomly $r < p$ and check $f(r, x_2, \ldots, x_n) = h(r)$
Completeness and Soundness

- If the prover is truthful, it can respond to queries with
  \[ h(z) = (1 - z) f(x_1, \ldots, z, \ldots, x_n) + z f(x_1, \ldots, z, \ldots, x_n). \]
- When \( n = 1 \), the verifier rejects an incorrect value of \( K \) with probability 1.
- If the prover is not truthful for some \( n > 2 \)
  - \( h(z) \neq (1 - z) f(x_1, \ldots, z, \ldots, x_n) + z f(x_1, \ldots, z, \ldots, x_n) \).
  - Both functions are linear, so they agree on at most one value of \( z \). The probability the verifier picks this as \( r \) is \( 1/p \).
  - The verifier rejects an incorrect \( h(z) \) with probability at least
    \[ P_n \leq (1 - 1/p)P_{n-1} \leq (1 - 1/p)^{n-1}. \]
- If \( p > 2n \), \( P_n \) is sufficiently small.
Summarizing IP

- TQBF is PSPACE-Complete
- TQBF is in IP with completeness 1
  - Recall that IP with arbitrary completeness is itself contained in PSPACE.
- It turns out IP remains the same even when random bits are public.
  - This is proven using hash functions. They allow a verifier to check if a set is at least size $|S|$ or at most size $|S|/2$.
- The provers for TQBF and GI need only be able to solve instances of TQBF and GI, respectively.
  - Using an interactive proof, a program solving these problems can be used to check itself.
MIP

- The power of interactive proofs can be increased with multiple provers (Ben-Or, Goldwasser, Kilian, Wigderson, 1988).
- For languages in MIP, the verifier can query two provers that cannot communicate.
  - The verifier can pit the provers against each other.
- Two provers can be used to recognize languages in IP in only a single round (Cai, Condon, Lipton, 1990)
- In fact, MIP = NEXPTIME (Babai, Fortnow, Lund, 1991)
  - Again, a single round of interaction suffices (Feige, 1991)
- Three or more provers can be simulated using 2 provers (Fortnow, Rompel, Sipser)
MIP and PCP

- MIP is incredibly powerful, but what if we keep questions short (O(log(n)))?
- The PCP theorem implies NP-Complete problems can be reduced to gap instances of 3SAT.
- We have short a two prover one round protocol:
  - Ask P₁ the assignment to the three variables in a clauses
  - Ask P₂ the assignment to one of the variables
  - Check for disagreement.
- Multiple repetitions will increase soundness.
  - What about asking multiple questions at once? Yes!
  - This is called parallel repetition (Raz, 1998)
Conclusion

- In addition to PCP and NP, IP and MIP show that interaction is a powerful tool in classifying problems.
- PSPACE is equivalent to a polynomial number of PTIME interactions with a prover.
- NEXPTIME is equivalent to a constant number of PTIME interactions with three provers or a polynomial number of interactions with two.
- Next time we use PCPs to implement a two prover 1 round interactive proof. This leads to strong hardness of approximation results.