CSCI 1590
Intro to Computational Complexity
Interactive Proofs

John E. Savage
Brown University

April 22, 2009
Summary

1. Interactive Proofs

2. Private versus Public Randomness

3. Bounding the Prover’s Resources
Interactive Proofs

- Last class we introduced the concept of an interactive proof, an interactive protocol in which a verifier, $V$, interacts with a prover, $P$.
- Given an input, $x$, $V$ is allowed to ask $P$ some number of questions. $V$ is a Turing machine (TM) with bounded computational resources, but $P$ can be an arbitrary function.
- A language, $L$, is recognized by an interactive proof with completeness $p_c$ and soundness $p_s$ if:
  - There exists a $P$ such that for all $x \in L$, $V$ accepts with probability at least $p_c$.
  - For all $P$, when $x \notin L$, $V$ rejects with probability at least $p_s$.
- When specifying an interactive proof for $L$, we describe $V$. To prove correctness, we assume $P$ is chosen to maximize the probability that $V$ accepts.
- We consider classes of languages recognized by interactive proofs where $V$ is a probabilistic TM that runs in polynomial time.
**Definition**

Let $\text{IP}[k]$ be the class of all languages, $L$, for which there exists a polynomial time PTM, $V$, that can decide whether or not $x \in L$ with completeness and soundness $2/3$, after $k$ total queries and responses to/from a prover, $P$. Let $\text{IP} = \bigcup_{c \geq 1} \text{IP}(n^c)$.

- In this definition, $P$ has unbounded computational resources. Later we show that $P$’s resources can in fact be bounded.
- Had we used a soundness requirement of 1 we would have $\text{IP} = \text{NP}$. Notice that this also holds if $V$ is deterministic.
  - When the soundness is 1, $V$ never accepts when $x \notin L$.
  - Any sequence of queries causing $V$ to accept acts as proof that $x \in L$.
  - In other words, when $x \in L$, we have a certificate that can be verified in polynomial time, but if $x \notin L$, no such certificate exists.
A pair of graphs \((G_1, G_2)\) is in \(GNI\) if they are not isomorphic.

\(GI\), the complement of \(GNI\), is in \(NP\), but not known to be \(NP\)-complete.

\(GNI \in IP[2]\). It can be recognized with completeness 1 and soundness 1/2 using a one round interactive proof

- A PTM, \(V\), randomly selects \(G_1\) or \(G_2\) and randomly permutes it.
- The randomly permuted graph, \(G'\), is given to a prover, \(P\), which is asked to return which graph \(V\) randomly selected.
- \(V\) rejects if the \(P\)'s answer is incorrect.

- The soundness of the proof can be increased by performing multiple queries in parallel. This is an example of “parallel repetition”.

- Although no resource limitation is placed on \(P\), it suffices for \(P\) to be able to solve arbitrary instances of \(GI \subseteq NP\).
In defining $\textbf{IP}[k]$, we have allowed the verifier to use a random string that remains hidden from the prover. Our interactive proof for $\text{GNI}$ relies on this fact.

It turns out that any “private-coin” protocol can be converted into a “public-coin” protocol with only a few additional rounds.

To do this for our $\text{GNI}$ protocol, we observe that our query can be rephrased as follows (where $\equiv$ denotes “is isomorphic”):

- Let $S_i = \{(H, \pi) : H \equiv G_i \text{ and } \pi(H) = G_i\}$.
- Let $n$ be the number of vertices in $G_i$. $|S_i| = n!$.
- Let $S = S_1 \cup S_2$. If $G_1 \equiv G_2$, $S_1 = S_2$ and $|S| = n!$. If $G_1 \not\equiv G_2$, $S_1 \cap S_2 = \emptyset$, so $|S| = 2n!$.

To give a public-coin protocol for $\text{GNI}$, we need a way for the verifier, $V$, to verify the size of $S$ using public random bits.
An Interactive Proof for Set Lower Bounds

- Let $k = \lceil \log_2 n! \rceil$. Notice that any element in set $S$ can be described using a unique binary string of length $2 \times k + 1$ bits. Furthermore if $P$ selects such a string, it is easy for $V$ to verify that it does correspond to an element of $S$.
- Now suppose that $V$ and $P$ have access to a randomly generated oracle, $f : \{0, 1\}^{2k+1} \mapsto \{1, 2, \ldots, 2n!\}$.
  - If $|S| = n!$, then given a random $(k + 1)$-bit string $r$, $P$ will be able to select an element $s \in S$ such that $f(s) = r$ with probability $1/2$.
  - If $|S| = 2n!$, then $P$ will be able to select an $s \in S$ such that $f(s) = r$ with probability at least $1 - (1 - 1/(2n!))^{2n!} \approx 1 - e^{-1}$
- Given $f$ and some public random coins, $V$ can verify the size of $S$ with high probability by repeatedly requesting $s \in S$ such that $f(s) = r$.
- To reduce the probability of error, $S$ can be replaced with a set of $t$-tuples, $S' = S \times S \times \ldots \times S$
- Generating a truly random $f$ requires an exponential number of random bits. Instead, $f$ can be selected from a properly chosen family of pairwise-independent hash functions.
The complexity class $\text{AM}[k]$ is the equivalent of $\text{IP}[k]$ when both prover and verifier have access to the same random bits.

The existence of a public-coin set lower bound protocol implies that $\text{GNI} \in \text{AM}[2]$ (also note, $\text{AM}[2] = \text{BP} \cdot \text{NP}$).

The same set lower bound protocol can be used to show that $\text{IP}[k] \subseteq [\text{AM} + 2]$. For intuition, notice that if $x \in L$, the set of random strings that cause $V$ to accept is large.

It is not hard to show that for all constants $k$, $\text{AM}[k] \subseteq \text{AM}[2]$. For this reason, $\text{AM}[2]$ is often denoted $\text{AM}$.

It is possible to modify the set lower bound protocol so that it has perfect completeness. This implies that $\text{AM}$ and $\text{IP}$ are unchanged even when defined using perfect completeness.
In defining $\text{IP}[k]$, $\text{AM}[k]$ and $\text{IP}$, we have allowed the prover, $P$, to be an arbitrary function. In fact, since the verifier, $V$, is a polynomial time PTM, we need only consider provers that require a polynomial amount of space.

To see why, recall that the provers “goal” in any interactive proof is to maximize the probability that $V$ accepts. Furthermore, since $V$ is fixed for a given proof, $P$ can be assumed to know $V$.

At any point in an interactive proof, when $V$ queries $P$, $P$ can compute its response by simulating all possible future exchanges between $V$ and $P$ to determine which response maximizes the probability that $V$ accepts. This takes exponential time, but only polynomial space.

Since both $P$ and $V$ can be simulated using a polynomial amount of space, $\text{IP} \subseteq \text{PSPACE}$.
More Practical Provers

- We can describe an interactive proof by describing a polynomial time PTM, $V$. To actually implement this protocol, however, we would need a valid prover $P$.

- Since this $P$ may need to perform an arbitrary $\text{PSPACE}$ computation, we do not appear to have an efficient implementation.

- To actually implement an interactive proof, we need $P$, as well as $V$, to be a PTM with polynomial runtime, but then what power does $P$ provide?

- Suppose we provide $P$ with access to a “secret” piece of information. If $V$ knew this secret, it would be as powerful as $P$, but as long as $V$ does not, $P$ can answer questions $P$ cannot (if $P \neq \text{NP}$).

- In a “zero-knowledge” interactive proof, $P$’s goal is to prove to $V$ some fact about a secret without revealing any information about it.
  - Examples: Do you have the right password? Is this graph 3 colorable?