

CSCI 1590
Intro to Computational
Complexity
Interactive Proofs

Eric Rachlin

Brown University

April 14, 2008

Summary

- 1 Randomized reductions
- 2 Two-Stage Proof Protocols
- 3 Interactive Proofs

Review of last lecture

- Last week we introduced probabilistic Turing machines (PTMs), Turing machines with access to random bits. We used these PTMs to define several complexity classes.
 - $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP} \subseteq \mathbf{BPP}$.
- To define these classes we identified two ways in which a PTM, M , can reasonably be said to recognize a language, L , efficiently:
 - M outputs the correct answer in polynomial time on average.
 - M always runs in polynomial time and is correct with high probability.
- In the second case, we distinguish between two directions of error:
 - When $x \in L$, there is a lower bound, $p_c > 1/2$, on the probability that M outputs true.
 - When $x \notin L$ there is a lower bound, $p_s > 1/2$, on the probability that M outputs false.
- The requirements associated with p_c and p_s are referred to as **completeness** and **soundness**, respectively.

Completeness and Soundness

- Languages in **ZPP** can be recognized in expected polynomial time with completeness 1 and soundness 1.
- Languages in **RP** can be recognized with completeness $2/3$ and soundness 1.
- languages in **BPP** can be recognized with completeness and soundness $2/3$.
- In the case of **BPP** and **RP**, we saw how error-reduction can be achieved by querying a PTM a polynomial number of times.
 - Using a Chernoff bound we saw that both completeness and soundness can be made exponentially close to 1 in n (the length of the input).
- Using Markov's inequality we showed that for classes defined with completeness or soundness less than 1, it is unnecessary to distinguish between expected runtime and worst-case runtime.

Probabilistic Reductions

- We can apply the notion of completeness and soundness not just to language recognition, but to resource-bounded reductions.
- Previously, we considered deterministic reductions from one language to another.
 - We defined **NP**-completeness, and **PSPACE**-completeness in terms of deterministic polynomial time reduction.
 - We defined **P**-completeness in terms of deterministic logarithmic space reduction.
- What if we allow our polynomial time reduction to have a one-sided? A two-sided error?

Two-Step Proof Protocols

- Let **BP · NP** be the class of languages, L that can be recognized using the following two-step protocol (with completeness and soundness $2/3$).
 - For some $L' \in \mathbf{NP}$, a PTM, V , performs a probabilistic polynomial time reduction, f , on its input x .
 - V accepts x if and only if $f(x) \in L'$.
 - If $x \in L$, V accepts with probability $\geq 2/3$, if $x \notin L$, V rejects with probability $\geq 2/3$.
- V is referred to as the “verifier”. The hypothetical entity providing V with proof that $f(x) \in L$ is called the “prover”.
- Let **RP · NP** be the class of languages defined by the above protocol using a soundness of 1.
- **RP · NP = NP**.
- **BP · NP** contains *GN1* (see next slide), a language not known to be in **NP**.
- It has been shown that if **BP · NP** contains a **coNP**-complete language such as $\overline{\text{SAT}}$, **PH** collapses.

Graph-Isomorphism

- Two graphs, G_1 and G_2 are **isomorphic** if there is a permutation (i.e. relabeling), π , of G_1 's vertices such that the two graphs are identical.
 - A pair of graphs, (G_1, G_2) , is in GI if and only if they are isomorphic.
 - A pair of graphs, (G_1, G_2) , is in GNI if and only if they are not isomorphic.
- $GI \in \mathbf{NP}$, but is not known to be **NP**-complete.
- GNI can be recognized with completeness 1 and soundness 1/2 using a simple three-step proof protocol
 - 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
- No resource limitation has been placed on the prover, but V runs in polynomial time.
- If the two graphs are not isomorphic, P can always determine which graph has been permuted. If they are isomorphic, however, either graph can be permuted to form G' , so all P can do is return a random guess.

Interactive Proofs

- The two- and three-step protocols we have described are all examples of **Interactive Proof**
- In an interactive proof, a verifier with bounded resources asks a series of questions to an all-powerful prover. The verifier must use the prover's answer to determine whether a string, x , is in a language, L .
 - In an interactive proof, a completeness requirement of p_c means that when $x \in L$, it is always possible for the prover to cause the verifier to accept with probability greater than p_c .
 - A soundness requirement of p_s means that when $x \notin L$, it is impossible for any prover to cause the verifier to accept with probability greater than p_s .
- Unless the soundness is less than 1, the languages recognized by an interactive proof are simply those in **NP**.
- If the verifier is deterministic, the languages recognized by an interactive proof are once again those in **NP**.

Definition

Let $\mathbf{IP}[k]$ 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 soundness and completeness $2/3$, after k rounds of the following protocol.

- Given x and all of P 's prior responses, V outputs a query y and sends it to a prover P .
- P , which has unbounded computational resources, and access to all prior queries, returns a response.

Let $\mathbf{IP} = \bigcup_{c \geq 1} \mathbf{IP}(n^c)$.

- To show that a language is in $\mathbf{IP}[k]$, we can describe a verifier V that executes k rounds of questions with a prover, P . We must then show that when $x \in L$, a P exists that can get V to accept with probability $\geq 2/3$, but when $x \notin L$, no P exists that can convince V to accept with probability $> 1/3$. V and P act as adversaries.

Some observations regarding **IP**

- In defining **IP**[k] and **IP**, we have allowed the prover, P , to be an arbitrary function. In fact, since the verifier, V , is a polynomial time PTM, it suffices to only consider provers that compute in polynomial space. We will use this fact when we show that **IP** = **PSPACE**.
- As we observed earlier, had we used a soundness requirement of $1/3$ (instead of $2/3$), **IP** = **NP**. In contrast, it is known that using a completeness requirement of 1 leaves **IP** unchanged.
- In defining **IP**[k], we have allowed the verifier to use random strings that remain hidden from the prover. It turns out that this requirement is unnecessary. We discuss this in detail next class.