Great Theoretical Ideas In Computer Science
Lecture 26
December 2, 2014

PROOFS REVISITED: Zero-knowledge, Interactive, Spot-checkable

Computational Lens on Proofs

Computer Science has led to significant new (and counterintuitive) perspectives on the age old concept of "proofs"

- Zero-knowledge proofs: Reveal nothing but truth of statement being proved!
- Interactive proofs: Easy to verify proofs for problems not believed to be in NP
- Probabilistically Checkable Proofs: Super easy to check by probing proof at just three random locations (Grader's dream)

Today's lecture: A glimpse into some of these ideas revolving around interactive proofs and probabilistic verification.

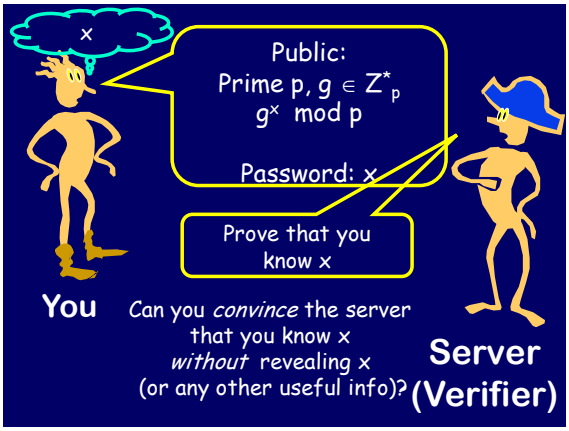
These are some of the most influential ideas in theoretical computer science in the last 25 years

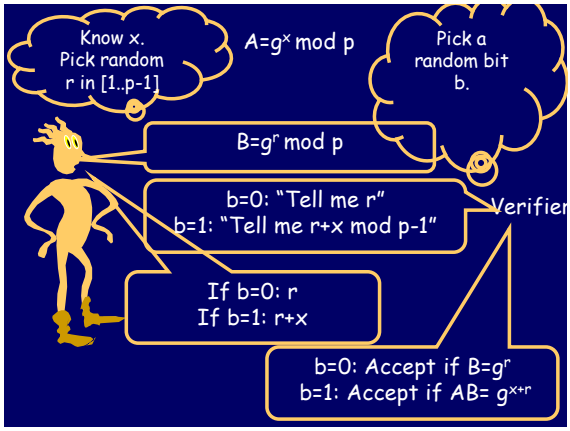
- Deep conceptual statements
- Many applications (cryptography, approximate optimization)

Traditional proofs:
1) Non-interactive
2) Can never prove false statements

The power of "proofs" enhanced by allowing:
1) Interaction between prover and verifier;
2) Allowing small probability that verifier accepts proof of a false statement

ZERO-KNOWLEDGE PROOFS: Proofs that reveal nothing beyond truth of statement being proved



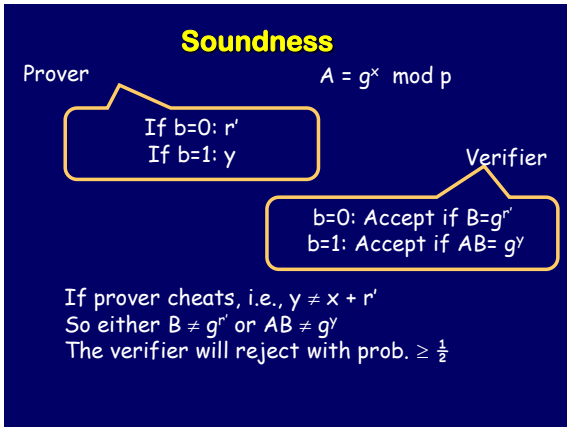


Completeness and soundness

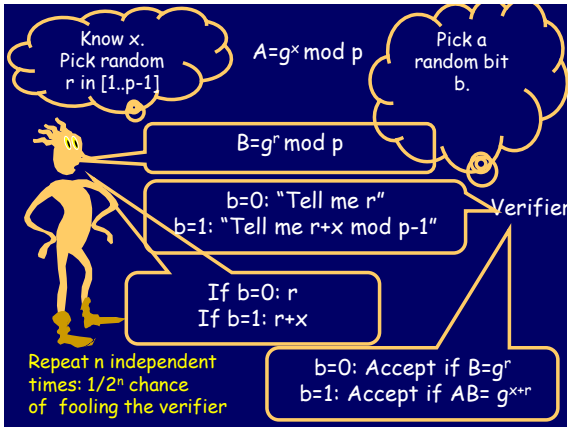
Completeness: If prover plays by the rules, then the verifier is convinced with probability 1.

Soundness: If prover cheats, the verifier rejects with probability $\frac{1}{2}$ in each iteration.

Soundness



If prover cheats, i.e., $y \neq x + r'$
So either $B \neq g^{r'}$ or $AB \neq g^{y}$
The verifier will reject with prob. $\geq \frac{1}{2}$



What does the verifier learn? (when prover is honest)

If b=0, he sees g^r for random r

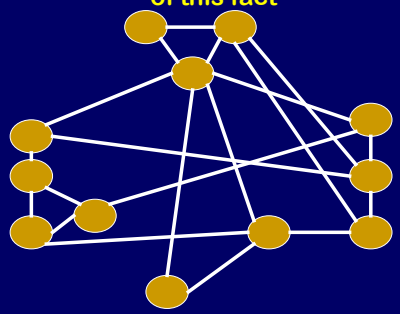
If b=1, he sees g^{x+r} for random r, i.e., again g^s for random s.

Verifier never sees both g^r and g^{x+r}

Verifier simply sees an independent random element of Z_p^* in each iteration.

- Not very useful, he could have generated these himself!

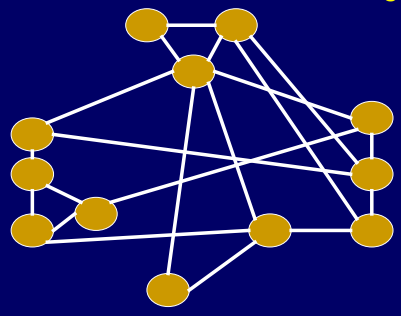
Prover knows a 3 coloring
Can he give a "Zero Knowledge" proof of this fact



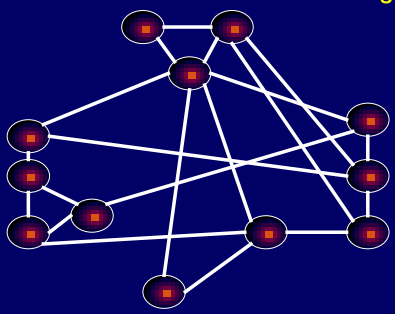
3! = 6 ways to permute the color names

Notice that for the secret coloring of graph, there are 6 colorings that can be obtained by permuting the 3 color names.

Prover randomly chooses one of the 6 colorings obtainable form the secret coloring.

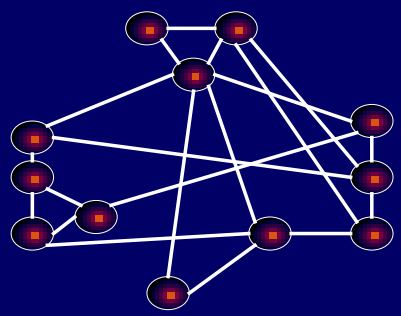


Prover randomly chooses one of the 6 colorings obtainable form the secret coloring.

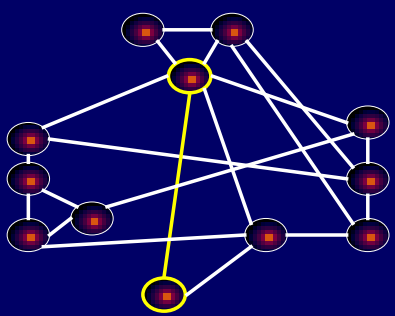


Each node will have an associated "locked box."The prover places the colors in the corresponding boxes.

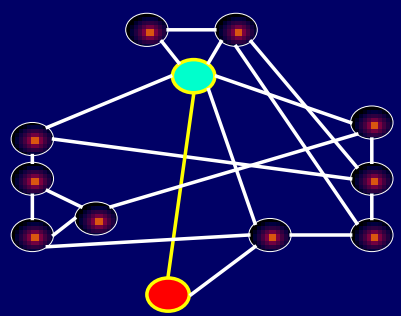
Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.



Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.

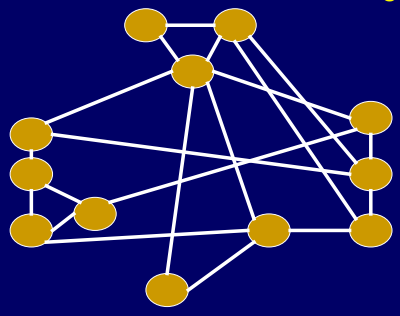


Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.

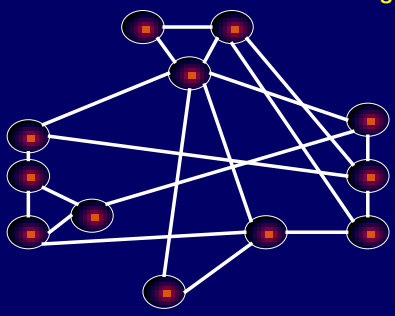


REPEAT USING AN INDEPENDENT, RANDOM CHOICE OF THE 6 PERMUTATIONS OF COLOR NAMES

Prover randomly chooses one of the 6 colorings obtainable form the secret coloring.

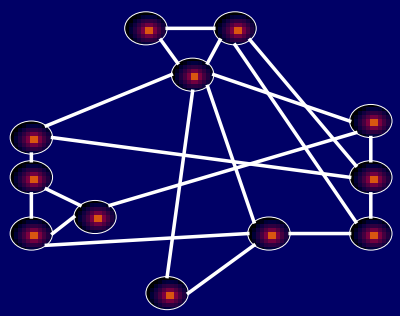


Prover randomly chooses one of the 6 colorings obtainable form the secret coloring.

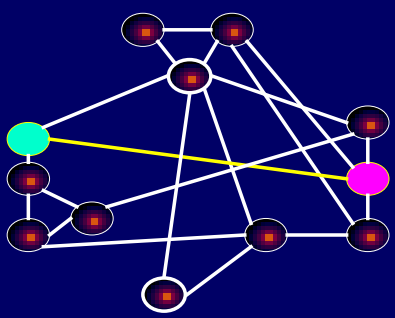


Each node will have an associated "locked box."The prover places the colors in the corresponding boxes.

Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.

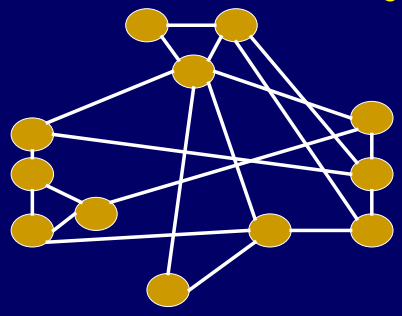


Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.

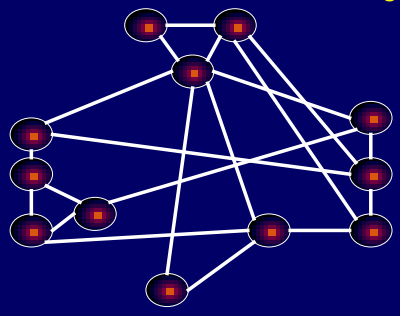


REPEAT USING AN INDEPENDENT, RANDOM CHOICE OF THE 6 PERMUTATIONS OF COLOR NAMES

Prover randomly chooses one of the 6 colorings obtainable form the secret coloring.

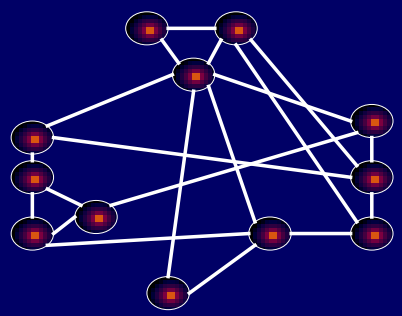


Prover randomly chooses of one the 6 coloring obtainable form the secret coloring.

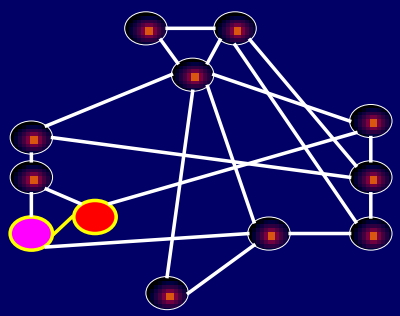


Each node will have an associated "locked box."The prover places the colors in the corresponding boxes.

Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.



Verifier: Picks an edge at random and asks to open the boxes at both ends of the edge.



Verifier repeats these independent challenges for cm where m = # edges in the graph

If ***all*** challenges result in the revelation of two different colors, the verifier accepts. Otherwise, the verifier rejects.

Verifier

Completeness: If graph is 3-colorable and prover has a valid coloring in mind, he can play by the rules and the verifier will accept.

Soundness: If graph is not 3-colorable, then for any coloring there will be an edge with the same color at each end, and Verifier has a 1/m chance of catching the prover's lie in each round.

Soundness

Probability of fooling the Verifier for all cm challenges $= (1 - 1/m)^{cm} < (1/e)^c$

Taking c=100, this is negligible

Zero-Knowledge (when prover plays by the rules)

In each iteration, Verifier sees colors of two adjacent vertices (based on an independent scramble of the color names):

- I.e., one of (1,2),(2,3),(3,1), (2,1),(3,2),(1,3) picked at random
- Nothing he didn't know/anticipate already!

Locked Boxes?

To make our arguments mathematical, we need to implement locked boxes with two properties:

1. Hiding: Verifier shouldn't be able to "open" the locks on the boxes and learn the coloring (this would compromise zero-knowledge)
2. Binding: Prover shouldn't be able to change the values in the boxes after he is challenged with an edge (this would compromise soundness)

Can be realized via "bit commitment schemes" (details skipped)

Binding property will hold for all prover strategies; Hiding property will hold for efficient verifiers.

ANY THEOREM CAN BE PROVED IN A ZERO KNOWLEDGE WAY!

Any proof of length m can be expressed as the 3-coloring of a graph of size poly(m) (follows from NP-completeness of 3COLOR)

You make a graph G such that everyone agrees that a 3-coloring of G is equivalent to a proof of the theorem.

You prove that you know a 3-coloring revealing zero information about anything else.

Interactive Proofs

So membership in NP languages can be proved in a "zero-knowledge" way.

Can we prove membership in languages beyond NP? (we'll drop zero-knowledge requirement for rest of the lecture)

Graph Isomorphism

Two graphs G=(V,E) and H=(W,F) are isomorphic if there exists a bijection $\sigma : V \to W$ such that $\forall a,b \in V$, $(a,b) \in E$ if and only if $(\sigma(a),\sigma(b)) \in F$

In such a case, we say $H = G\sigma$ (H is just G, relabeled)

Define the language
ISO = { (G,H) | G and H are isomorphic graphs}

ISO = {(G,H) | G and H are isomorphic}

ISO is not known to be in P.
(A major open question)

Is ISO $\in$ NP ?

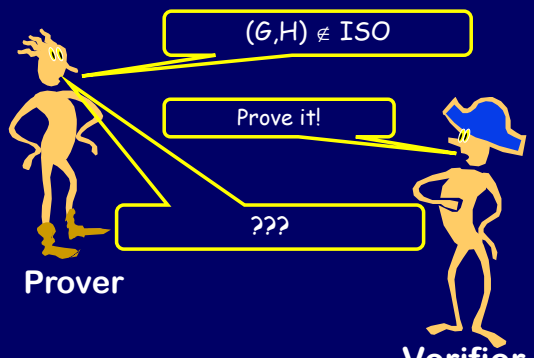
Sure, witness is just the bijection σ
Verifier can check that $H = \sigma G$

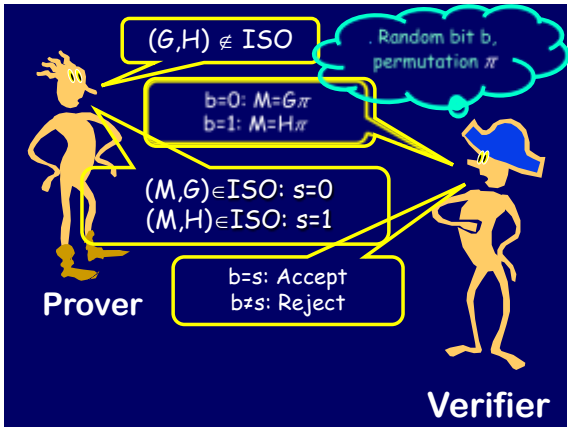
ISO = {(G,H) | G and H are isomorphic}
ISO ∈ NP

NON-ISO =
{(G,H) | G and H are **not** isomorphic}

We don't know if NON-ISO∈NP
(This would be a *great* PhD)

Turns out one with interaction, one can "prove" that two graphs are not isomorphic (convincing verifier with very high confidence)

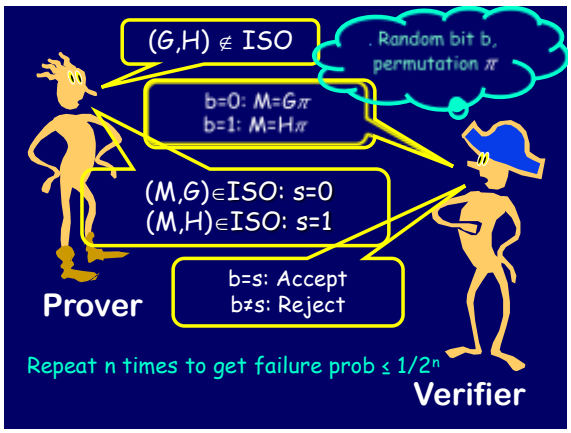




NON-ISO interactive protocol:
1. Verifier picks random b ∈ {0,1} and perm. π
   If b=0, he sends M= Gπ, else he sends M = Hπ
2. Prover responds with a bit s ∈ {0,1}
3. Verifier accepts iff b = s.

Completeness: If G and H are not isomorphic, M is isomorphic to exactly one of G,H, and the prover can figure out which graph was scrambled to produce M and make verifier always accept.

Soundness: If G and H are isomorphic, the distribution of M is independent of b. So Prover can't guess b with probability better than $\frac{1}{2}$. Thus Verifier accepts with probability $\leq \frac{1}{2}$



Prover any function P(x,c) = what to say next if c is conversation so far.

Verifier is a poly-time function V(x,r,c) = what to say next if c is conversation so far (r=verifier's random coins)

P<->V(x,r) denotes one conversation. Total bits in conversation can't exceed some poly(|x|). Verifier must accept/reject at end of conversation.

A language *L* belongs to IP iff

∃ polynomial time verifier V(x,r), |r|<poly(|x|); all V<->P(x,r) conversations are poly(|x|) bounded and accept/reject.

$x \in L$:
$\exists$Prover P $Pr_r[P(x) \leftrightarrow V(x,r)$ accepts$] = 1$

$x \notin L$:
$\forall$Prover P $Pr_r[P(x) \leftrightarrow V(x,r)$ accepts$] \leq 1/2$

NON-ISO belongs to IP

∃ polynomial time verifier V(x,r)
|r|<poly(|x|); all V<->P(x,r) conversations are poly(|x|) bounded and accept/reject.

$x \in L$: $\exists$Prover P $Pr_r[P(x) \leftrightarrow V(x,r)$ accepts$] = 1$
$x \notin L$: $\forall$Prover P $Pr_r[P(x) \leftrightarrow V(x,r)$ accepts$] \leq 1/2$

What else is in IP?

We have seen that NON-ISO has an interactive proof, even though we don't know if it is in NP.

What about problems we surely believe to not be in NP, such as complements of 3COLOR, 3SAT, or other NP-complete problems?

In particular, can one (interactively) prove that a 3SAT formula is NOT satisfiable?

A more general problem: #SAT How many satisfying assignments does φ have?

SAT = {φ | φ is a satisfiable CNF formula. }

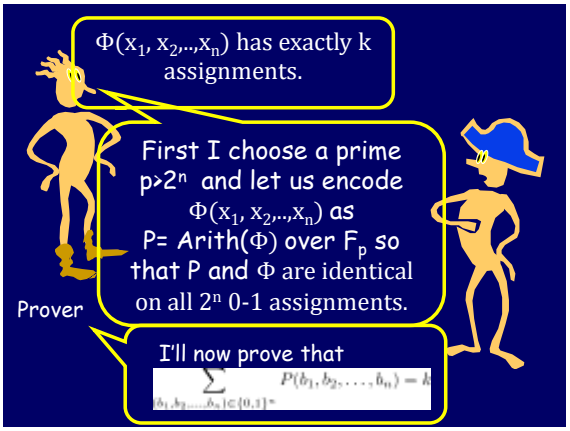
#SAT = {(φ,k) | φ has exactly k satisfying assignments}

Inductive Arithmetization: From Boolean formulae to a polynomial over a finite field.

Arith(T) = 1
Arith(F) = 0
Arith(not θ_1) = 1-Arith(θ_1)
Arith(θ_1 and θ_2) = Arith(θ_1) Arith(θ_2) ← note: $x^2=x$ so all powers are 0 or 1

Inductive invariant:
θ and Arith(θ) agree on all 0/1 assignments.

Also, degree(Arith(θ)) = O(size(θ))



Let's define some polynomials

$P(x_1, x_2, \ldots, x_n) = \text{Arith}(\phi)$; n-variate polynomial

$$P_0 = \sum_{b_1,\ldots,b_n \in \{0,1\}} P(b_1, b_2, \ldots, b_n)\text{; To prove } P_0 = k$$

$$P_1(x) = \sum_{b_2,\ldots,b_n \in \{0,1\}} P(x, b_2, \ldots, b_n)\text{; Univariate polynomial in } F_p[x]$$

Prover can send $P_1(x)$ and verifier can check $P_1(0) + P_1(1) = k$.

But what if prover sends bogus $P_1(x)$?

To combat this, Verifier picks random $r_1 \in F_p$, and challenges prover to "prove" the value of $P_1(r_1)$

Key point: If prover sends polynomial $Q_1(x) \neq P_1(x)$, then $Q_1(r_1) \neq P_1(r_1)$ with prob. $\geq$ 1- $O(\text{size}(\phi)/p)$

The power of polynomials

Two n-variable formulae ϕ and ψ can differ on just one out of 2^n assignments. So can't catch their difference by checking at a random assignment.

Two low-degree polynomials $P \neq Q$ must differ on significant fraction of the domain.

This property was very useful for "error-correction" and is now handy again.

Amazing reach of this simple fact about polynomials: a nonzero degree d polynomial has at most d roots over a field.

$P(x_1, x_2, \ldots, x_n) = \text{Arith}(\phi)$; n-variate polynomial

$$P_1(x) = \sum_{b_2,\ldots,b_n \in \{0,1\}} P(x, b_2, \ldots, b_n)\text{; Univariate polynomial in } F_p[x]$$

Prover sends low-degre polynomial $P_1(x)$ (by listing its coefficients); verifier checks $P_1(0) + P_1(1) = k$, picks random $r_1 \in F_p$, and challenges prover to prove the claimed value of $P_1(r_1)$.

Lies beget lies: If prover sends poly $Q_1(x) \neq P_1(x)$, then $Q_1(r_1) \neq P_1(r_1)$ with high probability.

Next round:

$$P_2(x) = \sum_{b_3,\ldots,b_n \in \{0,1\}} P(r_1, x, b_3, \ldots, b_n)\text{; Univariate poly. in } F_p[x]$$

Prover sends low-degre polynomial $P_2(x)$ (by listing its coefficients); verifier checks $P_2(0) + P_2(1) = P_1(r_1)$, picks random $r_2 \in F_p$, and challenges prover to prove the claimed value of $P_2(r_2)$.

Next round:

$$P_2(x) = \sum_{b_3,\ldots,b_n \in \{0,1\}} P(r_1, x, b_3, \ldots, b_n)\text{; Univariate poly. in } F_p[x]$$

Prover sends low-degre polynomial $P_2(x)$ (by listing its coefficients); verifier checks $P_2(0) + P_2(1) = P_1(r_1)$, picks random $r_2 \in F_p$, and challenges prover to prove the claimed value of $P_2(r_2)$.

Lies beget more lies: If prover sends polynomial $Q_2(x) \neq P_2(x)$, then $Q_2(r_2) \neq P_2(r_2)$ with high prob.

Round i invariant: Verifier has chosen $r_1, r_2, \ldots, r_{i-1}$. Prover must commit to a polynomial, which for honest behavior should be

$$P_i(x) = \sum_{b_{i+1},\ldots,b_n \in \{0,1\}} P(r_1, \ldots, r_{i-1}, x, b_{i+1}, \ldots, b_n)$$

Verifier checks $P_i(0) + P_i(1) = P_{i-1}(r_{i-1})$, picks random $r_i \in F_p$, and tasks prover with backing up the claimed value of $P_i(r_i)$.

Final round: Verifier has chosen $r_1, r_2, \ldots, r_{n-1}$. Prover sends a univariate low-degree polynomial, supposedly

$$P_n(x) = P(r_1, \ldots, r_{n-1}, x)$$

Verifier checks $P_n(0) + P_n(1) = P_{n-1}(r_{n-1})$, picks random $r_n \in F_p$, and checks $P_n(r_n) = P(r_1, r_2, \ldots, r_n)$.

Verifier rejects if any of its checks across the n rounds fails; otherwise he accepts.

Completeness: If $\Phi(x_1,x_2,..,x_n)$ has exactly k assignments, then a prover playing honestly by the rules will satisfy all checks made by the verifier, and the verifier will accept with certainty.

Soundness Theorem: If number of satisfying assignments to $\Phi(x_1,x_2,..,x_n)$ doesn't equal k, then the verifier accepts with probability $\leq$ poly(n)/p << 1/2

Proof idea: Let $Q_i(x)$ be poly. prover sends in round i
Since # sat. assignments to $\Phi = P_1(0) + P_1(1) \neq k$,
prover must lie about $P_1(x)$ in round 1, sending $Q_1 \neq P_1$.
(otherwise the check $Q_1(0)+Q_1(1)=k$ will fail)

Now $P_2(0)+P_2(1)=P_1(r_1)$ (by defn) & $P_1(r_1) \neq Q_1(r_1)$ w.h.p.
So prover is forced to lie in round 2, sending $Q_2 \neq P_2$
(otherwise the check $Q_2(0)+Q_2(1)=Q_1(r_1)$ will fail)

Continuing this argument, unless very lucky in an earlier round, prover must send $Q_n(x) \neq P_n(x)$ in round n.
Verifier can compute $P_n(r_n) = P(r_1,r_2, \ldots, r_n)$
(as he knows P) & will find $P_n(r_n) \neq Q_n(r_n)$ w.h.p.

Probability of accepting false claim

For verifier to accept, prover must get lucky in some round.

Let i be the earliest round where this happens, i.e., $P_i(r_i) = Q_i(r_i)$ even though $P_i(x) \neq Q_i(x)$

As P_i and Q_i are degree poly(n) polys, this happens with probability $\leq$ poly(n)/p

The probability that prover gets lucky in *some* round is at most n times bigger, and thus also $\leq$ poly(n)/p

Summary

One can prove that a 3SAT formula is not satisfiable via an interactive proof!
(Note: verifier is efficient, prover has to work hard)

Via NP-completeness reductions, same Holds for claim that graph is not 3-colorable, not Hamiltonian, etc.

In fact, IP=PSPACE.
The power of interactive proofs extends to all problems solvable in polynomial *space*

Probabilistically Checkable Proofs (PCP)

Aka proofs for the lazy (busy?) grader

Back to NP proofs

Let $L \in$ NP, say 3COLOR.

Traditional NP proof: A 3-coloring (O(n) bits for n-node graph).

Verifier has to read full proof, And check that each edge is colored with two different colors.

Can one write a proof that the verifier can "spot check" at random locations and catch errors with good confidence?)

Probabilistically Checkable Proofs

proof π of length poly(n)

Verifier V (randomized Polytime)

Reads O(1) random locations (independent of proof length)

Completeness:
If $G \in$ 3COLOR,
$\exists \pi$ that V accepts with prob. 1

Soundness:
If $G \notin$ 3COLOR,
$\forall \pi$, V accepts with prob. $\leq \frac{1}{2}$

PCP theorem

In fact, verifier can get by with just 3 queries!

PCP Theorem (1992) is one of the crowning achievements of CS theory (made the NYT)

"There is a format of writing proofs of theorems such that the proof can be verified with good confidence by only probing 3 randomly chosen (from an appropriate distribution) locations."

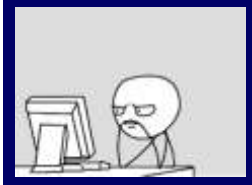
Proof overhead (of strongest known forms):
n poly(log n) if original witness length = n.

PCP theorem ingredients

Proof is half-a-semester+ course.
Blends together:

- P/NP
- expanding graphs
- random walks
- polynomials/finite fields
- error-correcting codes
- Fourier analysis

Illustrates coolness of theoretical CS:
Many interesting Math topics get mixed together.
Can work on whatever you want.



Zero-knowledge proofs
(intuitive argument
for zero-knowledge)

Interactive proofs beyond NP
(soundness arguments)

Study Guide

This is my last lecture this semester.
THANK YOU !!