15-251: Great Theoretical Ideas in Computer Science

Fall 2016 Lecture 29

December 8, 2016

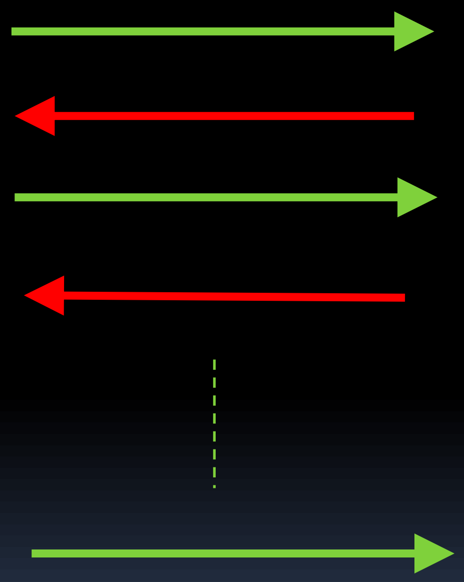
Epilogue:
Interactive Proof for $\overline{3SAT}$ and Couple of open problems

Interactive Proofs

Claim $x \in L$

Prover
(all powerful)

Verifier
(randomized
polynomial time)



poly($|x|$) long
conversation

Accept or
reject

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

There is a polynomial time verifier V(x,r), |r|<poly(|x|); all P↔V(x,r) conversations are poly(|x|) bounded and accept/reject.

- Completeness
  - x∈L: ∃ Prover P, Pr_r[P↔V(x,r) accepts] $\geq \frac{3}{4}$
- Soundness
  - x∉L: ∀ Provers P, Pr_r[P↔V(x,r) accepts] $\leq \frac{1}{4}$

We saw last lecture that graph non-isomorphism 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?

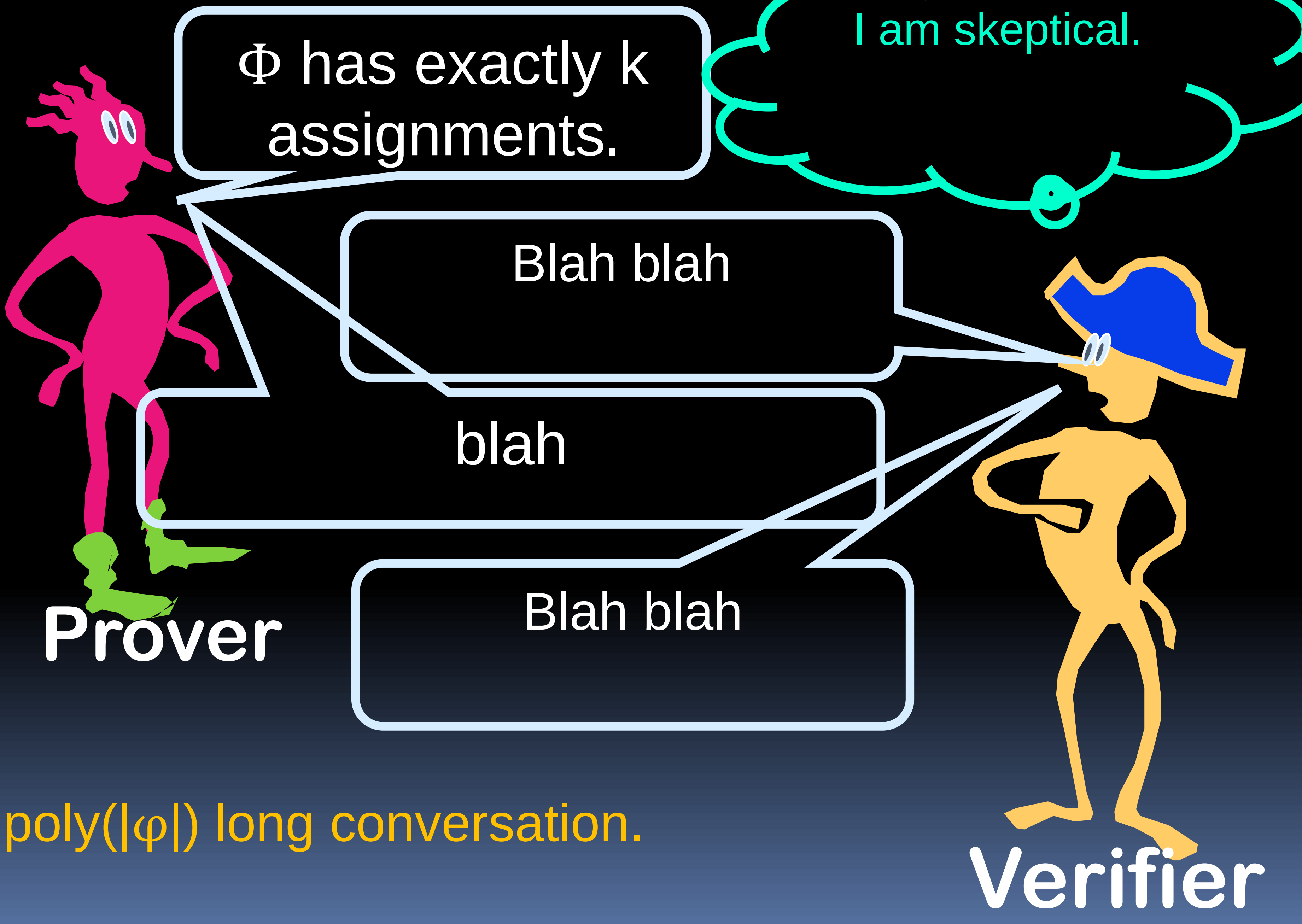
In particular, can one (interactively) prove that a 3SAT formula is NOT satisfiable?

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

3SAT = {φ | φ is a satisfiable 3-CNF formula. }

#3SAT = {(φ,k) | φ is a 3CNF formula that has exactly k satisfying assignments}

#3SAT ∈ IP



poly(|φ|) long conversation.

A Great Theoretical Idea

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

- Arith(T) = 1, Arith(F) = 0
- Arith(x) = x
- $\text{Arith}(\neg\theta_1) = 1\text{-Arith}(\theta_1)$
- $\text{Arith}(\theta_1 \wedge \theta_2) = \text{Arith}(\theta_1)\ \text{Arith}(\theta_2)$
- $\text{Arith}(\theta_1 \vee \theta_2) = 1 - (1\text{- Arith}(\theta_1))(1\text{-Arith}(\theta_2))$

Claim (proof by induction): For any Boolean formula θ, θ and Arith(θ) agree on all 0/1 assignments.

Also, degree(Arith(θ)) = O(size(θ))

Example

Suppose

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4)$$

$\text{Arith}(\phi) =$

$$\big(1 - (1 - x_1)x_2(1 - x_3)\big) \cdot \big(1\ - x_1 x_3(1 - x_4)\big) \cdot (1 - x_2 x_4)$$

Product of m terms if there are m clauses.
Each has degree at most 3 if ϕ is a 3SAT instance

Claim: $\Phi(x_1, x_2,..,x_n)$ has exactly k assignments.

First I choose a prime $p>2^n$ and let us encode $\Phi(x_1, x_2,..,x_n)$ as $P=$ Arith(Φ) over F_p so that P and Φ are identical on all 2^n 0/1 assignments.

Prover

I'll now prove that

$$\sum_{(b_1,b_2,\ldots,b_n)\in\{0,1\}^n} P(b_1, b_2, \ldots, b_n) = k$$

Let's define some polynomials

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

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

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

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(size(ϕ)/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.

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

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

Prover asked to send low-degree 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 asked to send 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 $\ll$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 $P(r_1,r_2, \ldots, r_n) \neq Q_n(r_n)$ 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, power of interactive proofs extends to all problems solvable in polynomial *space* (IP=PSPACE)

Surprising efficacy of polynomials in unexpected places!!

Problem 1: Bin Packing

Given a set $A = \{a_1, a_2, \ldots, a_n\}$ of positive integers, and a positive integer $B \geq \max a_i$, partition A into minimum number of subsets such that the sum of the elements in each subset is at most B.

Midterm 2: Bin Packing is NP-hard
(reduction from PARITITION to telling if two "bins" suffice)

Open: Is there a polytime algo to find a partition with OPT + 1 subsets, where OPT is the number of subsets in an optimal solution?

Best known: (1+ϵ) OPT + 1, or OPT + O(log OPT)

Problem 2: Graph 3-Coloring

3COLOR = { <G> | G is 3-colorable} is NP-complete.

So, unless P=NP, there is no polynomial time algorithm that given as input a 3-colorable graph, finds a proper 3-coloring of it.

Also known (and harder to prove):
Finding a 4-coloring of a 3-colorable graph is NP-hard.

Open: What about 5-coloring a 3-colorable graph?

Most likely NP-hard, but we can’t prove it!

Best algorithm uses $\approx n^{0.2}$ colors, where $n =$ #vertices

Problem 3: Finding a satisfying assignment when there is an abundance of them

Suppose we are given a CNF formula ϕ on n variables that is promised to have $\geq 2^{n-1}$ satisfying assignments (i.e., at least ½ the assignments satisfy every clause).

- Trivial to find a satisfying assignment of such an instance in randomized polynomial time.

<u>Open</u>: Is there a *deterministic* polynomial time algorithm for finding a satisfying assignment given such an instance of SAT?

Best known runtime: n^{O((log log n)^2)} (2016)