Proof Systems:

Note Title 5/14/2013

given language L : goal to prove $x \in L$.

proof system for L requires a Verification algorithm V

Completeness: $x \in L \Rightarrow \exists\ proof_x$ V accepts $(x, proof_x)$

$x \notin L \Rightarrow \forall\ proof'$ V rejects $(x, proof')$.

The prover asserts $x \in L$

true assertions have proofs

false assertions have no proofs.

Efficiency: $\forall x$, proof $V(x, proof)$ runs in time $poly(|x|)$.

$L \in NP$ iff $L = \{x \mid \exists y\ |y| \le |x|^k\ (x,y) \in R\}$ $R \in P$.

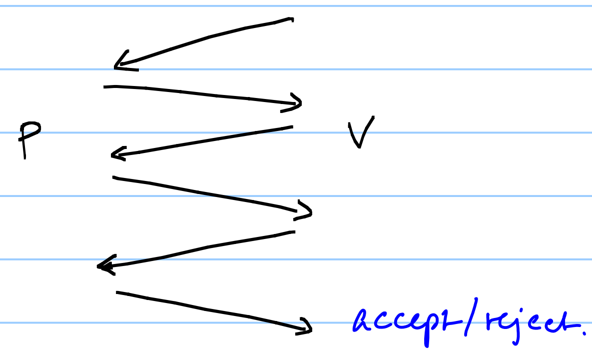
Verifier $= R$, proof is y.

New ingredients: randomness (verifier can toss coins)

interaction - instead of reading the proof, the verifier can ask questions.

Interactive proof systems:

both P (prover) and V (verifier) know x.



rounds $\le poly(|x|)$

V can use a random string.

An Interactive Proof System for L is an interactive protocol (P, V)

Completeness : $x \in L \Rightarrow \Pr[V \text{ accepts in } (P,V)(x)] \geq 2/3$.
Soundness : $x \notin L \Rightarrow \forall P^* \Pr[V \text{ accepts in } (P^*,V)(x)] \leq 1$

V is a p.p.t. machine.

as usual, repetition can reduce the error to ϵ.

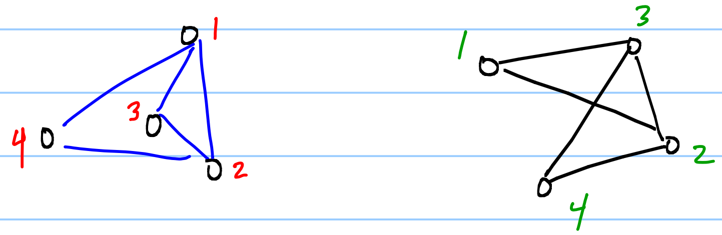
$IP = \{L \mid L \text{ has an interactive proof system}\}$.

$\Rightarrow$ philosophically captures more broadly what it means to be convinced that a statement is true.

$NP \subseteq IP$: verifier receives only a single message and uses no random bits.

If $NP \subsetneq IP$ then randomness is essential.
if the verifier is deterministic, the prover knows all of V's questions in advance and can send all the answers in one shot.

Graph Isomorphism : $G_0 = (V, E_0)$ $G_1 = (V, E_1)$
graphs are isomorphic $G_0 \cong G_1$ iff $\exists \pi : V \to V$.
$(x,y) \in E_0 \iff (\pi(x), \pi(y)) \in E_1$



$1 \to 3$
$3 \to 1$
$4 \to 4$
$2 \to 2$

$GI = \{(G_0, G_1) \mid G_0 \cong G_1\}$

$GI \in NP$ but not known to be in P or NP-complete.

$GNI = \overline{GI}$ Not known if $GNI \in NP$.

Theorem $GNI \in IP$. (indication that IP may be more powerful than NP).

Verifier		Prover
Flips coin $c \in \{0,1\}$ pick random π Apply π to G_c	$\xrightarrow{H = \pi(G_c)}$	if $H \cong G_0$ $r = 0$ Else $r = 1$
Accept iff $r = c$.	$\longleftarrow$	

Completeness: if $G_0 \not\cong G_1$ then $H \cong G_0$ or $H \cong G_1$ but not both. Prover can select the correct one.

Soundness: if $G_0 \cong G_1$, then prover sees the same distribution regardless of $c = 0, 1$.
Prover gets no information about c.
Any prover can succeed w.p. $\leq 1/2$.

$GNI \subseteq$ co-NP but not known if GNI is co-NP-complete.

Theorem co-NP $\subseteq$ IP.

Proof idea. Will actually show the following language is in IP:
$\{ (\phi, k) \mid \phi(x_1 \ldots x_n)$ has exactly k SAT assignments$\}$

Prover claims that ϕ has exactly k satisfying assignments.

this is true iff:

$\phi(0, x_2, .., x_n)$ has k_0 sat assignments.

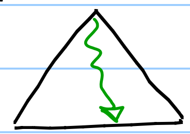
$\phi(1, x_2, .., x_n)$ has k_1 sat assignments.

$k = k_0 + k_1$.

Prover sends k_0 and k_1.

then Verifier picks a random c and asks prover to recursively prove that $\phi = \phi(c, x_2 \cdots x_n)$ has k_c satisfying assignments.

At the end, verifier can check.



2^n leaves $\leftrightarrow$ assign.

Problem: If k is only off by 1, verifier will only catch this if V's random choices lead exactly to the leaf that is the source of the discrepancy.

Solution: Replace $\{0,1\}^n$ w/ $\{F_q\}^n$

$\hookrightarrow$ integers mod q.

Verifier substitutes a random field element at each step. Vast majority of field elements at each step will catch a cheating prover (instead of just one).

(LFKN)

Theorem: $L = \{ (\phi, k) \mid$ CNF ϕ has exactly k satisfying assignments $\}$.

① Arithmetization: $\phi(x_1, \ldots, x_n) \Rightarrow P_\phi(x_1 \ldots, x_n)$

degree d polynomial over F_q q prime $> 2^n$

$x_i \longrightarrow x_i$
$\phi \wedge \phi' \rightarrow \quad P_\phi \cdot P_{\phi'}$
$\phi \vee \phi' \rightarrow \quad 1 - (1 - P_\phi)(1 - P_{\phi'})$

For all $x \in \{0,1\}^n \qquad P_\phi(x) = \phi(x)$
degree d of $P_\phi \leq |\phi|$.
Can compute $P_\phi(x)$ in poly time, given ϕ & x.

Prover wishes to show: $k = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} P_\phi(x_1 \cdots x_n)$

Define $k_z = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} P_\phi(z, x_2, \ldots, x_n)$

Prover sends k_z for all $z \in \mathbb{F}_q$

Verifier picks random z
- asks prover to prove that $k_z = \sum_{x_2} \cdots \sum_{x_n} P_\phi(z, x_2 \ldots, x_n)$
- also checks that $k_0 + k_1 = k$.

At end, verifier checks $P(z_1, z_2 \ldots, z_n) = k_n$.

actually ... Since $q > 2^n$, prover can't send all of them.
Sends the polynomial $p(z) = \sum_{x_2} \cdots \sum_{x_n} P_\phi(z, x_2 \ldots, x_n)$
degree $\leq d \leq |\phi|$.

Protocol shown on next page...

Input (ϕ, k)

Prover | **Verifier**

$P_1(x) = \sum_{x_2 \cdots x_n} P_\phi(x, x_2, \ldots, x_n)$ $\longrightarrow$ $P_1(0) + P_1(1) = k$?

all $x_i \in \{0,1\}$

pick random $z_1 \in \mathbb{F}_q$

$\xleftarrow{z_1}$

$P_2(x) = \sum_{x_3 \cdots x_n} P_\phi(z_1, x, x_3 \cdots x_n)$ $\xrightarrow{P_2(x)}$ $P_2(0) + P_2(1) = P_1(z_1)$?

pick random $z_2 \in \mathbb{F}_q$

$\xleftarrow{z_2}$

$P_3(x) = \sum_{x_4 \cdots x_n} P_\phi(z_1 z_2, x, x_4 \cdots x_n)$ $\xrightarrow{P_3(x)}$

$\vdots$

$\xrightarrow{P_n(x)}$ $P_n(0) + P_n(1) = P_{n-1}(z_{n-1})$

pick random $z_n \in \mathbb{F}_q$

$P_n(z_n) = P_\phi(z_1, \ldots, z_n)$

Completeness: $(\phi, k) \in L$ honest prover will always cause the verifier to accept.

Soundness: $P_1(x)$ is correct poly. $P_1^*(x)$ poly sent by prover.

Suppose $(\phi, k) \notin L \Rightarrow P_1(0) + P_1(1) \neq k$

if $P_1^*(0) + P_1^*(1) \neq k \Rightarrow V$ rejects.

So what if $P_1^*(0) + P_1^*(1) = k$?

(it must be $P_1^* \neq P_1$).

Bad case: Verifier picks z_1 s.t. $P_1(z_1) = P_1^*(z_1)$

$\Pr_z[P_1^*(z_1) = P_1(z_1)] \leq d/q \leq |\phi|/2^n$.

Suppose we don't have the bad event:

$$p_1^*(z_1) \neq p_1(z_1)$$

Value Prover is claiming to verifier. ↑ true value.

Verifier sends $p_2^*(x)$ if $p_2^*(0) + p_2^*(1) \neq p_1^*(z_1) \Rightarrow$ reject.

o.w. $p_2^*(0) + p_2^*(1) = p_1^*(z_1) \neq p_1(z_1) = p_2(0) + p_2(1)$

$\Rightarrow p_2^* \neq p_2$.

if $p_2(z_2) = p_2^*(z_2) \rightarrow$ BAD!

In general: Bad event: $p_i(z_i) = p_i^*(z_i)$

Probability any bad event occurs: $\leq n \dfrac{|\phi|}{2^n} <$ small.

At the end, we have $p_n^*(z)$ which the prover claims is equal to $p_n^*(z) = p_\phi(\underbrace{z_1, \ldots, z_{n-1}}_{\text{fixed}}, \underbrace{z}_{\text{variable}})$ if no "bad" event occurs, these polynomials are actually different.

Verifier selects a random z_n and checks.

$$p_n^*(z_n) = p_\phi(z_1, \ldots, z_n)$$

Fails to detect difference w/ prob $\leq \frac{|\phi|}{2^n}$.

If no bad even occurs, verifier will detect the difference:

=

$L = \{ (\phi, k) \mid$ CNF ϕ has exactly k satisfying assignments $\}$ is in IP

L is coNP-hard so coNP $\subseteq$ IP.

NP, co-NP $\subseteq$ IP. How much more is in IP.

Theorem (Shamir) IP = PSPACE.

IP $\subseteq$ PSPACE is easy: enumerate all possible interactions, explicitly calculate acceptance probability.

Interaction very powerful!
(Can interact with master player of Generalized Geography and determine if she can win, even if you can not compute the optimal moves).

Need to prove PSPACE $\subseteq$ IP $\rightarrow$ (i.e. QSAT $\in$ IP).
Same basic idea as co-NP proof - plus a few additional ingredients.

First assume no occurrance of x_i separated by more than one $\forall$ from point of quantification: Condition (*).

$$\exists x \underbrace{\exists \ \forall \ \exists}_{\text{occurrences of } x} \forall \underbrace{\qquad\qquad}_{\text{no occurrences of } x}$$

This helps ensure degree of any single variable $O(|\phi|)$

Arithmetize $\phi \rightarrow P_\phi$

$$\exists x_i \ \phi \rightarrow \sum_{x_i=1,0} P_\phi(x_i \ldots)$$

$$\forall x_i \ \phi \rightarrow \prod_{x_i=0,1} P_\phi(x_i, \ldots).$$

this can double the degree and square the size of P.

$\Rightarrow$ Quantified Boolean expression ϕ is true iff $P_\phi > 0$.

Problem: the $\forall$ (Π) terms, may cause $P_\phi > 2^{2^{|\phi|}}$ ← too large to compute.

Solution: evaluate mod q $\quad 2^n < q < 2^{3n}$.

→ Prover sends a "good" q in the first round

a "good" q is one such that $P_\phi \bmod q > 0$.

Claim: a good q exists because the # primes in the range is at least 2^n.

↳ the product of all these primes is $> P_\phi$, so it can't be that $P_\phi \bmod q = 0$ for all the q in the range.

Since the prover is trying to get V to accept, he is motivated to send a good q.

QSAT protocol:

Input ϕ → Prover, Verifier

Prover		Verifier
$k, q, P_1(x)$ $P_1(x)$: remove outer Σ or Π from P_ϕ	$\xrightarrow{k, q, P_1(x)}$	$P_1(0) + P_1(1) = k$ or $P_1(0) \cdot P_1(1) = k$] depending on $\forall$ or $\exists$ pick random $z_1 \in \mathbb{F}_q$
	$\xleftarrow{z_1}$	
$P_2(x)$ remove outer Σ or Π from $P_\phi[x_1 \leftarrow z_1]$	$\xrightarrow{P_2(x)}$	Check $P_2(0) + P_2(1) = P_1(z_1)$ or $P_2(0) P_2(1) = P_1(z_1)$ pick random $z_2 \in \mathbb{F}_q$
	$\xleftarrow{z_2}$	
$\vdots$		
	$\xrightarrow{P_n(x)}$	$P_n(0) + P_n(1) = P_{n-1}(z_{n-1})$? or $P_n(0) P_n(1) = P_{n-1}(z_{n-1})$? pick random $z_n \in \mathbb{F}_q$ Check $P_n(z_n) = P_\phi[x_1 \leftarrow z_1 \cdots x_n \leftarrow z_n]$

Note: because of condition (*) $p_n(x)$ is bounded in size, so it can be sent to V.

Also, $p_\phi[x_1 \leftarrow z_1, \ldots, x_n \leftarrow z_n]$ has no quantifiers since all the x_i are fixed so this can be computed in poly time. //

Completeness: if $\phi \in$ SAT then an honest prover will cause V to accept.
↳ must also send good q.

Soundness: Let $p_i(x)$ be correct polynomial
Let $p_i^*(x)$ be polynomial sent by prover.

If $\phi \notin$ QSAT ⇒ then $p_1(0) +/* \, p_1(1) = 0 \neq k$.
If $p_1^*(0) +/* \, p_1^*(1) \neq k$ then V will reject
If this doesn't happen then $p_1^* \neq p_1$.

$$\text{Prob}_{z_1}[\, p_1^*(z_1) = p_1(z_1)\,] \leq \frac{2|\phi|}{2^n}$$

← this because of (*).

In general: assume $p_i(z_i) \neq p_i^*(z_i) = p_{i+1}(0) +/* \, p_{i+1}(1)$.
If $p_{i+1}^*(0) +/* \, p_{i+1}^*(1) \neq p_i^*(z_i)$ then V rejects.
otherwise $p_{i+1}^* \neq p_{i+1}$

$$\Rightarrow \text{Prob}_{z_{i+1}}[\, p_{i+1}(z_{i+1}) = p_{i+1}^*(z_{i+1})\,] \leq \frac{2|\phi|}{2^n}$$

At the end, we have $p_n(z_n) \neq p_n^*(z_n)$
↑ Can calculate this from the original ϕ, given $z_1 \cdots z_n$

V will detect they are unequal.

V will reject as long as $p_i(z_i) \neq p_i^*(z_i)$ $\forall i$.

For each i, the probability $p_i(z_i) = p_i^*(z_i) \leq \frac{2|\phi|}{2^n}$.

Prob that for any i, $p_i(z_i) = p_i^*(z_i) \leq \frac{2n|\phi|}{2^n} << 1/3$.

$\Rightarrow$ QSAT is in IP.

Example: $\phi = \forall x \exists y (x \vee y) \wedge \forall z ((x \wedge z) \vee (y \wedge \neg z))$
$\vee \exists w (z \vee (y \wedge \neg w))$

$$P_\phi = \prod_{x=0,1} \sum_{y=0,1} \left[(x+y) * \prod_{z=0,1} \left[(xz + y(1-z)) + \sum_{w=0,1} (z + y(1-w)) \right] \right]$$

$$\prod_{x=0,1} \sum_{y=0,1} \left[(x+y) * \prod_{z=0,1} \left[(xz + y(1-z)) + 2z + y \right] \right]$$

$$\prod_{x=0,1} \sum_{y=0,1} \left[(x+y) * (2y)(x+2+y) \right]$$

$$\prod_{x=0,1} (x+1)(2)(x+3) = (1 \cdot 2 \cdot 3)(2 \cdot 2 \cdot 4) = 96. = P_\phi$$

Prover claims $P_\phi > 0$.

Prover sends $q = 13$; claim $P_\phi = 96 \bmod 13 = 5$; sends $k = 5$.

Prover removes outer $\prod_{x=0,1}$ and sends $(x+1)2(x+3) = 2x^2 + 8x + 6 = p_1(x)$

Verifier checks $p_1(0)p_1(1) = (6)(16) = 96 \equiv 5 \bmod 13$.

Verifier picks randomly $z_1 = 9$.

$$P_\phi[x_1 \leftarrow 9] = \sum_{y=0,1} \left[(9+y) * \prod_{z=0,1} \left[(9z + y(1-z)) + \sum_{w=0,1} (z + y(1-w)) \right] \right]$$

$$p_2(y) = (9+y) * \prod_{z=0,1} \left[(9z + y(1-z)) + 2z + y \right]$$

$$= (9+y)(2y)(11+y) = 2y^3 + 18y^2 + 22y^2 + 198y \pmod{13}$$

$$= 2y^3 + y^2 + 3y$$

Prover sends $p_2(y) = 2y^3 + y^2 + 3y$

Verifier checks $\underbrace{p_2(0) + p_2(1)}_{\equiv 6} = \underbrace{p_1(9)}_{\equiv 6}$

Picks random $z_2 = 3$ $\quad p_2(3) = 7$

$$\| $$

$$P_\phi[x \leftarrow 9, y \leftarrow 3] = \left[\underbrace{(9+3)} * \underbrace{\prod_{z=0,1} \left[(9z + 3(1-z)) + \sum_{w=0,1} (z + 3(1-w)) \right]} \right]$$

$$P_3(z) = (9z + 3(1-z)) + (2z+3) = 8z + 6.$$

$\Rightarrow$ $12 \cdot P_3(0) \cdot P_3(1)$ should be $P_2(3)$

Prover + verifier both know $12(\ldots)$.

Prover sends $p_3(z)$.

Verifier checks $12 \cdot p_3(0) \cdot p_3(1) = 12 \cdot 14 \cdot 6 \bmod 13 = 7$ ✓

Verifier picks random $z_3 = 7$.

$$P_\phi[x \leftarrow 9, y \leftarrow 3, z \leftarrow 7] = 12 * \left[(\overbrace{(9 \times 7) + 3(1-7)}^{6 \bmod 13.}) + \sum_{w=0,1} (7 + 3(1-w)) \right]$$

$$=$$

$$12 * P_3(7) = 12 * 10$$

everyone agrees on $12 * [6 + (\text{—})]$ prover sends $\overset{P_4(w)}{10w + 10}$

verifier checks $12 * [6 + (10 + 20)] = 12 * 10 \bmod 13.$ ✓

verifier picks random $z_4 = 2$.

Final check: $12\left[(9 \cdot 7) + 3(1-7) + (7 + 3(1-2))\right] = 12\left[6 + p_4(2)\right]$ ✓

<u>Arthur-Merlin Games</u>: IP permits verifier to keep coin flips private... is this necessary?

Note that the GNI protocol breaks without it.

Arthur-Merlin game: interactive protocol in which coin flips are public (although not known in advance).

Arthur may just as well send the results of his coin flips and ask Merlin to do whatever computation he would have done.

Clearly Arthur-Merlin $\subset$ IP.

"private coins are at least as powerful as public coins."

The proof of IP = PSPACE: actually shows. → results of random z choices are revealed.

PSPACE $\subseteq$ Arthur Merlin $\subseteq$ IP $\subseteq$ PSPACE

"public coins are at least as powerful as private coins!".

Limiting # of rounds: AM[k] #rounds = k, Arthur (V) goes first.
MA[k] # rounds = k, Merlin (P) goes first.

<u>Theorem</u>: AM[k] = AM[k] with perfect completeness
MA[k] = MA[k] with perfect completeness

↪ $x \in L \Rightarrow$ accept w.p. 1.

Theorem: $\forall k \geq 2 \quad AM[k] = AM[2]$.

Will Show: $MA[2] \subset AM[2]$.

then can move all of Arthurs messages to beginning.

$$AMAMAM \subseteq AAMMAM \cdots \subseteq AAAMMM$$

Proof: given $L \in MA[2]$. → m chosen, then r [MA]

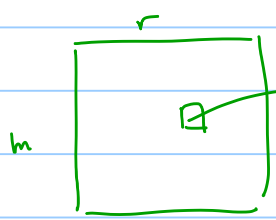
$$x \in L \Rightarrow \exists m \; \Pr_r[(x,m,r) \in R] = 1.$$

$$\Pr_r[\exists m \; (x,m,r) \in R] = 1.$$

→ r chosen, then m. [AM].

$$x \notin L \Rightarrow \forall m \; \Pr_r[(x,m,r) \in R] \leq \epsilon.$$

$$\Rightarrow \Pr_r[\exists m \; (x,m,r) \in R] \leq 2^{|m|}\epsilon.$$



$(x,m,r) \in R$?

the # of "yes" boxes in any row $\leq 2^{|r|}\epsilon$.

the # cols that have a "yes" $\leq 2^{|m|}2^{|r|}\epsilon$.

$\Rightarrow$ Prob a random col has any "yes" $\leq 2^{|m|}\epsilon$.

by t repetitions, can get $\epsilon < 2^{-t}$

repeat w/ t independent random strings: $t = m+1$

$$2^{|m|}\epsilon < 1/2.$$

Two important classes: $MA = MA[2]$

$AM = AM[2]$.

Definitions w/o reference to interaction:

$L \in MA$ iff $\exists$ poly time language R:

$$x \in L \Rightarrow \exists m \; Pr_r[(x,m,r) \in R] = 1$$
$$\forall m \; Pr_r[(x,m,r) \in R] \le 1/2$$

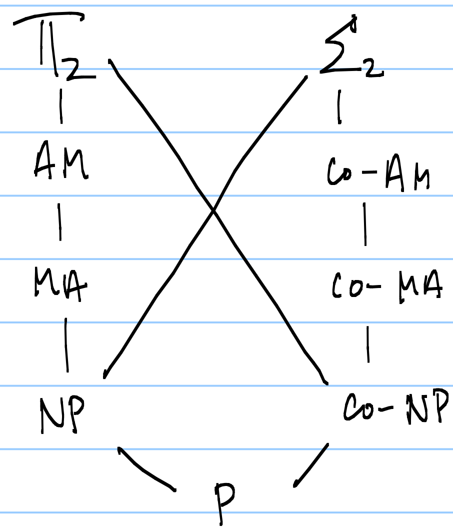
$L \in AM$ iff $\exists$ poly-time language R:

$$x \in L \Rightarrow Pr_r[\exists m \; (x,m,r) \in R] = 1.$$
$$Pr_r[\exists m \; (x,m,r) \in R] \le 1/2.$$

$AM, MA \subseteq NP$ (can elect to ignore r).

$AM, MA \subseteq \Pi_2$ $\qquad$ $L \in AM$: $x \in L \iff \forall r \exists m \; (x,m,r) \in R.$

$$\underbrace{MA \subseteq AM}_{\text{from above.}} \subseteq \Pi_2$$



Theorem $\text{co-NP} \subset AM \Rightarrow PH = AM.$

Suffices to show: $\Sigma_2 \subset AM$

because then $\Sigma_2 \subset \underbrace{AM \subset \Pi_2}_{\text{from before.}}$

we showed earlier that this causes $PH = \Sigma_2.$

Proof: $\quad \text{co-NP} \subset AM \Rightarrow \Sigma_2 \subset AM.$

$L \in \Sigma_2 \Rightarrow \exists$ poly-time R

$$x \in L \Rightarrow \exists y \forall z \; (x,y,z) \in R.$$
$$x \notin L \Rightarrow \forall y \exists z \; (x,y,z) \notin R.$$

Merlin sends y

1 AM round decides co-NP query $\forall z\, (x,y,z) \in R$?

3 rounds: MAM since $MA \subseteq AM$

$\Rightarrow AMM \subseteq AM$.

We know Arthur-Merlin $= IP$

Theorem $IP[k] \subset AM[O(k)]$.

implies $\forall k \geq 2$ $IP[k] = AM[O(k)] = AM[2]$

$GNI \in IP[2] = AM$

Not known if GI is NP-complete.

If it is GNI is co-NP-complete.

$\Rightarrow$ co-NP $\subset AM \Rightarrow AM = PH$ (unlikely!).