Today

- Power of the prover: IP $\subseteq$ PSPACE.
- IP[poly] $\subseteq$ AM[poly].
- IP[k] $\subseteq$ AM[k].
- Start IP $=$ PSPACE.

The optimal prover

- Given a fixed verifier, what should a prover do?
- Can figure out what to do, optimally, by computing the following quantity:
- Given a history of interactions so far, what is the highest probability, over all provers, of the verifier accepting.
- Can compute this by induction on number of remaining rounds.
- Prover that does this is the optimal prover.

IP $\subseteq$ PSPACE

Simple consequence of the explicit form of the optimal prover:

Proposition: IP $\subseteq$ PSPACE.

Proof: Can compute "probability of acceptance by optimal responses" in PSPACE.

IP[poly] $=$ AM[poly]

- Lets draw the interaction tree:
  - Nodes correspond to history so far: questions asked and optimal answers.
  - Edges between history and its immediate successor.
- Assume w.l.o.g. that questions are all binary, and given a path to leaf, there is a unique random string leading to this path (achieved by verifier announcing its random string after the protocol).
- Label leaves as accept/reject.
- Label node with # accepting leaves.

- Verifier's goal: Verify label of root is at least $2/3\times$ # random strings.

IP[poly] = AM[poly]

- Starting with root, and going down some path in the tree, Arthur repeats the following:
- Inductively, Arthur has a lower bound L_u on label of current node u. Arthur asks prover for optimal answers to two children, and labels L_v and L_w of corresponding nodes v and w.
- Arthur verifies $L_w + L_v \geq L_u$. Verifies v with probability $L_v/(L_v + L_w)$ and w otherwise.
- At root $L_{\text{root}} = 2/3\times$ # random strings. At leaf, verify verdict is accepting.

IP[poly] = AM[poly]

Let N_u denote actual # accepting leaves in subtree.

- Claim: Prob. verifier given at u accepts $\leq N_u/L_u$.
- Claim: If $N_{\text{root}} \geq 2/3\times$ # random strings, then setting $L_u = N_u$ for every u, gives proof that is accepted with probability 1.
- Thm: IP[poly] = AM[poly]-one-sided.

(Theorem above due to [Goldwasser-Sipser] + [Furer-Goldreich-Mansour-Sipser-Zachos]. Proof due to [Kilian].)

AM proof for approximate set size

Suppose $S \subseteq \{0,1\}^n$ has size either $|S| \geq$BIG $= 2^m$ or at most $SMALL = 2^m/100$, where e.g., $m = \sqrt{n}$. Further $x \in S$? can be determined by Arthur on its own.

Can Merlin convince Arthur that S is BIG?

[Goldwasser-Sipser] give AM protocol for above.

Goldwasser-Sipser protocol

Protocol: (reminiscent of Sipser-Lautemann)

- Merlin picks (random) hash function $h : \{0,1\}^n \to \{0,1\}^{m-4}$. and sends to verifier.
- Arthur picks $y \in \{0,1\}^{m-4}$ at random and sends to Merlin.
- Merlin responds with $x \in S$ such that $h(x) = y$.

Goldwasser-Sipser protocol

Claim: If h is chosen from a nice p.w.i. family of hash functions, and $|S| \geq 2^m$, then for $2/3$ of y's, there exists $x \in S$ such that $h(x) = y$.

Claim: If $|S| \leq 2^m/100$, then no matter which h we pick, at most $16/100 \leq 1/6$ for the y's have $x \in S$ such that $h(x) = y$.

IP[k] ⊆ AM[k]

Will only prove IP[1] ⊆ AM[O(1)]. Extension to general k similar.

- Fix verifier with completeness $2/3$, and soundness $1/\text{poly}$.
- Let Q be set of possible questions.
- For $q \in Q$, let S_q be set of random strings that lead to question q being asked, where optimal prover leads to acceptance.
- Let r be length of random strings.
- So either $\sum_{q \in Q} |S_q| \geq (2/3)2^r$, $\sum_{q \in Q} |S_q| \leq 1/\text{poly}(r)$.

- For simplicity assume $|S_q| = 0$ or 2^l for every q.
- Will run two G-S protocols back to back.
- Will ask Merlin to prove $\#q$ such that $|S_q| = 2^l$ is at least $(2/3)2^{r-l}$.
- To do so, Merlin send h, Arthur queries with y and Merlin sends $q \in Q$ such that $h(q) = y$.
- Arthur still needs to verify $|S_q| \geq 2^l$. Does this with another G-S protocol.
- Working out details .... get theorem.

One-sided error?

Can get one-sided error protocols using more ideas from Lautemann-Sipser. (Pick many hash functions; one of them always has a pre-image.)

Corollary: Can prove graph non-isomorphism without error or private coins! Can you come up with elementary protocol?

#P $\subseteq$ PSPACE

Still don't have a way for Merlin to convince Arthur that there's so seating for the round-table!

Will work towards that today.

Not so far from Kilian's proof .... Just one more trick!

Arithmetizing SAT

Literal polynomials: $x \mapsto x$, $\overline{x} \mapsto (1-x)$.

Clause polynomial: $C(x,y,z)$ converted to $P(x,y,z)$; $x \vee y \vee z \mapsto 1-(1-x)(1-y)(1-z)$.

SAT polynomial: $\phi(x_1,\ldots,x_n) \rightarrow Q(x_1,\ldots,x_n)$ where $Q(\mathbf{x}) = \prod_{i=1}^m P_i(\mathbf{x})$ if $\phi = \wedge_{i=1}^m C_i$.

Property $Q(x_1,\ldots,x_n)$: for $\mathbf{a} \in \{0,1\}^n$, $Q(\mathbf{a}) = 1$ if $\mathbf{a}$ satisfies ϕ and 0 otherwise.

Q is a polynomial of degree m in each variable.

$\#\phi = \sum_{\mathbf{a}\in\{0,1\}^n} Q(\mathbf{a})$.

#SAT tree & Q-tree

Draw tree of Q-values:

Root = value of $\sum_{\mathbf{a}\in\{0,1\}^n} Q(\mathbf{a})$.

Node = value of sum on suffix, with prefix set to some fixed value.

$Q_{\mathbf{b}} = \sum_{\mathbf{c}\in\{0,1\}^?} Q(\mathbf{b},\mathbf{c})$.

Verifier verifies $Q_{\mathbf{b}} = Q_{\mathbf{b}0} + Q_{\mathbf{b}1}$.

Now need to to verify $Q_{\mathbf{b}0}$ and $Q_{\mathbf{b}1}$.

Can't afford to do this!

#SAT in IP

Will arbitrarily consider $Q_{\mathbf{b}}$ for every $\mathbf{b} \in \mathbb{Z}_p^?$ for some prime p.

What meaning does it have? None seemingly, but $Q_{\mathbf{b}}$ is well defined!

Suppose prover claims $Q_\lambda = \#\phi = N$. Will ask prover to prover $Q_\lambda = N(\mod p)$.

IP protocol for #SAT

Recursively Arthur is verifying: $Q_{\mathbf{b}} = K(\mod p)$.

Consider the function $p_{\mathbf{b}}(x) = \sum_{\mathbf{c} \in \{0,1\}^?} Q(\mathbf{b}, x, \mathbf{c})$

$p_{\mathbf{b}}$ is a univariate polynomial of degree m.

Arthur asks Merlin for $p_{\mathbf{b}}(x)$.

Merlin responds with $h(x)$.

Arthur verifies $h(0) + h(1) = K$.

Arthur picks random $\alpha \in \mathbb{Z}_p$ and sends to Merlin,

Now recursively verify $Q_{\mathbf{b}\alpha} = h(\alpha)$.

At end Arthur can compute verify $Q_{\mathbf{b}} = K$, since $Q_{\mathbf{b}} = Q(\mathbf{b})$.

Soundness

Completeness obvious.

For soundness, will claim:

Claim: If $Q_{\mathbf{b}} \neq K$, then $\Pr_\alpha[Q_{\mathbf{b}\alpha} = h(\alpha) \& h(0) + h(1) = K] \leq m/p$.

Theorem follows (modulo details).