Today

- $\#P \subseteq IP$.
- Polynomial straightline programs and interactive proofs.
- Straightline programs for PSPACE.

Recall #P, IP

- #P is the class of counting functions. Prototypical example: #SAT - # of satisfying assignments of a 3CNF formula.
- IP is the class of languages with interactive proofs. So far know that IP contains NP and GNI (graph nonisomorphism).
- Anything else? Today will show #P has interactive proofs. Also try showing PSPACE has IP.
- Remarks:
  - Need to use multiple rounds of interaction (so not an "AM" proof system.).

  - PSPACE is closed under complement. Any reason to believe IP is?

Basic Idea

- Suppose Prover wishes to prove ϕ has A satisfying assignments.
- Can use self-reducibility:
  - Can prove $\phi|_{x_1=0}$ has A_0 assigments and $\phi|_{x_1=1}$ has A_1 assigments, and that $A_0 + A_1 = A$.
  - Unfortunately # statements to be proved is growing exponentially.
  - Any way to commit to $\#\phi_{x_1=0}$ and $\#\phi_{x_1=1}$ jointly and then prove only one claim?
  - How does $\#\phi_{x_1=\alpha}$ behave as a function of α - naturally?

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

Proof: CRT to get initialization right over p. Schwartz Lemma for inductive step.

Theorem follows (modulo details).

Abstracting the proof

- Proof uses very little specific to #P.
- More about "downward self-reducibility and polynomials".
- Specifically, downward self-reducibility leads to the tree.
- Algebra compresses questions down to one question.
- In fact, don't need any structure on the questions!

Extending compression: Low-degree curves

Suppose computing $Q_{\mathbf{b}}(\mathbf{x})$ involves computing $Q_{\mathbf{c}}(\mathbf{y})$ and $Q_{\mathbf{c}}(\mathbf{z})$, where $\mathbf{y}$ and $\mathbf{z}$ are not related. Can we extend our idea to this case?

Lines in $\mathbb{F}^n$: $\ell : \mathbb{F} \to \mathbb{F}^n$.

Geometrically - a line is a line.

Algebraically: it is a collection of n functions, each of which is a degree 1 polynomial.

For any two points $\mathbf{y}$ and $\mathbf{z}$, there is a line ℓ s.t. $\ell(0) = \mathbf{y}$ and $\ell(1) = \mathbf{z}$. Specifically $\ell(t) = (1-t)\mathbf{y} + t\mathbf{z}$.

Why are lines nice?

$Q \circ \ell : \mathbb{F} \to \mathbb{F}$ is a polynomial of (at most) same degree as Q.

Extending the protocol's capabilities

- At ith level, to verify $Q(\mathbf{x}) = a$, the verifier generates $\mathbf{y}$ and $\mathbf{z}$ and ℓ containing $\mathbf{y}$ and $\mathbf{z}$. Asks prover for $Q \circ \ell$.
- Prover responds with degree d univariate polynomial h.
- Verifier verifies consistency assuming h is right, and then verifies $h(\alpha)$ is correct for random α.

Straightline program of polynomials

Defn: $p_0, \ldots, p_L$ is an (n, d, L, w)-straight line program of polynomials if

- Every p_i is on at most n variables.
- Every p_i is of degree at most d.
- p_i is constructed from p_{i-1} in a simple form. (Formally, there is a polynomial time algorithm A that, given i, $\mathbf{x}$ and an oracle for p_{i-1} can compute $p_i(x)$ making at most w non-adaptive queries to p_i.)
- p_0 is computable in polynomial time.

Polynomial program satisfiability

Defn: Polynomial straight line program polynomial satisfaction is the language whose instances are $(\mathbf{x}, a, \langle p_0, \ldots, p_L \rangle)$ such that $p_L(\mathbf{x}) = a$, where $\mathbf{x} \in \mathbb{Z}^n$, $\mathbf{a} \in \mathbb{Z}$ and $p_0, \ldots, p_L$ is an (n, d, L, w)-straightline program of polynomials.

Polynomial program is in IP for $w = 2$

Verifier runs in time $\mathrm{poly}(n, d, L, \log ||\mathbf{x}||)$.

- Verifier picks random prime $p \approx \mathrm{poly}(n, d, L, \log ||\mathbf{x}||)$ and sends to prover. Sets $a_L \leftarrow a$, and $\mathbf{x}_L \leftarrow \mathbf{x}$.
- For $i = L - 1$ downto 0 do:
  - Let $\mathbf{y}_i$ and $\mathbf{z}_i$ be queries of A on input $i + 1$, $\mathbf{x}_{i+1}$. Let ℓ_i be line thru $\mathbf{y}_i$ and $\mathbf{z}_i$. Verifier asks prover for $p_i \circ \ell_i$. Prover responds with h_i.
  - Verifier verifies that A's answer on oracle values $h(0)$ and $h(1)$ is a_{i+1}.
  - Verifier picks random $\alpha \in \mathbb{Z}_p$ and sets $\mathbf{x}_i \leftarrow \ell_i(\alpha)$ and $a_i \leftarrow h_i(\alpha)$.

- At end verifier verifies $h_0(\alpha) = p_0(\ell_0(\alpha))$.

Completeness $= 1$.

Soundness $\leq \ell d/p +$ CRT.

Poly program sat. is PSPACE complete

- Basic idea: $f_i(\mathbf{a}, \mathbf{b})$ has configurations $\mathbf{a}$ and $\mathbf{b}$ as inputs (if from $\{0,1\}^s$), and $f_i(\mathbf{a}, \mathbf{b}) = 1$ if get from $\mathbf{a}$ to $\mathbf{b}$ in exactly 2^i steps.
- f_0 is a constant-degree polynomial, of degree C in each variable.
- $f_{i+1}(\mathbf{a}, \mathbf{b}) = \sum_{\mathbf{c} \in \{0,1\}^s} f_i(\mathbf{a}, \mathbf{c}) f_i(\mathbf{c}, \mathbf{b})$ is also a polynomial of degree C in each variable.
- Unfortunately $w \neq 2$.
- Can fix easily: Will do summation slowly.

PSPACE-completeness

Define longer sequence:

- $g_i = g_{is} = f_i$.
- $g_{i0}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = g_{i-1,s}(\mathbf{a}, \mathbf{c}) \cdot g_{i-1,s}(\mathbf{c}, \mathbf{b})$.
- $g_{ij}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = g_{i,j-1}(\mathbf{a}, \mathbf{b}, \mathbf{c}0) + g_{i,j-1}(\mathbf{a}, \mathbf{b}, \mathbf{c}1)$, where $\mathbf{c} \in \mathbb{Z}_p^{s-j}$.
- g has degree at most C in the variables of $\mathbf{a}$, $\mathbf{b}$, and at most $2C$ in the variables of $\mathbf{c}$.
- $g_0, g10, g_{11}, \ldots, g_{1s}, g_{20}, \ldots, g_{ss}$ is a sequence of width $w = 2$.
- PSPACE completeness follows.

Conclusion

- PSPACE complete problem (Poly. program sat.) has an IP.
- PSPACE $\subseteq$ IP.
- Can generalize lines argument even "wider", for $w > 2$.
- Exercise: Do this, and thus give direct proof that the permanent has an interactive proof, where the prover only needs to be able to compute permanent.