Shamir's Theorem

**Theorem:** $\text{IP} = \text{PSPACE}$

- Note: $\text{IP} \subseteq \text{PSPACE}$
  - enumerate all possible interactions, explicitly calculate acceptance probability
  - interaction extremely powerful!
  - An implication: you can interact with master player of Generalized Geography and determine if she can win from the current configuration even if you do not have the power to compute optimal moves!

Shamir's Theorem

- need to prove $\text{PSPACE} \subseteq \text{IP}$
  - use same protocol as for $\text{coNP}$
  - some modifications needed
**The QSAT protocol**

<table>
<thead>
<tr>
<th>Prover</th>
<th>input: $\phi$</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_k(x)$: remove outer $\Sigma$ or $\Pi$ from $p_k$</td>
<td>$p_k(0)p_k(1) = k^2$ or $p_k(0)p_k(1) = -k^2$</td>
<td>$p_k(0)p_k(1)p_k(z_i) \neq 0$ or $p_k(0)p_k(1)p_k(z_i) = 0$</td>
</tr>
<tr>
<td>$p_k(x)$: remove outer $\Sigma$ or $\Pi$ from $p_k(x)$</td>
<td>pick random $z_i$ in $F_4$</td>
<td>pick random $z_i$ in $F_4$</td>
</tr>
<tr>
<td>$p_k(x)$: remove outer $\Sigma$ or $\Pi$ from $p_k(x)$</td>
<td>$p_k(0)p_k(1)p_k(z_i) \neq 0$ or $p_k(0)p_k(1)p_k(z_i) = 0$</td>
<td>$p_k(x) = p_k(x_{-2},..x_{-k})$</td>
</tr>
</tbody>
</table>

**Analysis of the QSAT protocol**

- **Completeness:**
  - if $\phi \in \text{QSAT}$ then honest prover on previous slide will always cause verifier to accept

- **Soundness:**
  - let $p_i(x)$ be the correct polynomials
  - let $p_i^*(x)$ be the polynomials sent by (cheating) prover
  - $\phi \in \text{QSAT}$ $\Rightarrow p_i^*(0) = p_i(1) \neq k$ $\Rightarrow \phi$ is "simple"
  - either $p_i^*(0) = p_i^*(1) \neq k$ (and $V$ rejects)
  - or $p_i^* \neq p_i$ $\Rightarrow Pr_{F_4}[p_i^*(z_i) = p_i(z_i)] \leq 2k/2^n$
  - assume $(p_{i+1}(0) + x_1 p_{i+1}(1) = p_i(z_i) \neq p_i^*(z))$
  - either $p_{i+1}^*(0) = p_{i+1}^*(1) \neq p_i^*(z)$ (and $V$ rejects)
  - or $p_{i+1} \neq p_{i+1}$ $\Rightarrow Pr_{F_4}[p_i^*(z_i) = p_i^*(z_i)] \leq 2k/2^n$

- **Conclusion:** QSAT is in IP

**Example**

- Papadimitriou – pp. 475-480

\[ \phi = \forall x \exists y (x \land y) \land \forall z ((x \land z) \lor (y \land \neg z)) \lor \exists w (z \land (y \land \neg w)) \]

\[ P_\phi = \prod_{i=0}^{n}(x_i + y_i) \prod_{i=0}^{n}(z_i + (1-z_i)) + \prod_{i=0}^{n}(z_i + y_i) \]

$P_\phi = 96$ but $V$ doesn’t know that yet!

**Analysis of protocol**

- **Soundness (continued):**
  - if verifier does not reject, there must be some $i$ for which:
    - $p_i^* \neq p_i$ and yet $p_i^*(z_i) = p_i(z_i)$
    - for each $i$, probability is $\leq 2^{-i}p_{2^n}$
    - union bound: probability that there exists an $i$ for which the bad event occurs is $\leq 2n(2^n) = \text{poly}(n)/2^n << 1/3$
  - conclude: QSAT is in IP

**Example**

$P_\phi = \prod_{i=0}^{n}(x_i + y_i) \prod_{i=0}^{n}(z_i + (1-z_i)) + \prod_{i=0}^{n}(z_i + y_i) \]

Round 1: (prover claims $p_0 > 0$)
- prover sends $q = 13$; claims $p_0 = 96$ mod 13 = 5; sends $k = 5$
- prover removes outermost $\lceil \rceil$; sends
  \[ p_1(x) = 2x^2 + 8x + 6 \]
- verifier checks:
  - $p_1(0)p_1(1) = (6)(16) = 96 = 5$ (mod 13)
  - verifier picks randomly: $z_1 = 9$
Example
φ = ∀x∃y(x ∨ y) ∧ ∃z((x ∧ z) ∨ (y ∧ ¬z)) ∨ ∃w(z ∨ (y ∧ ¬w))

Round 2: (prover claims this = 6)
- prover removes outermost \( \Sigma \); sends
  \( p_2(y) = 2y^2 + y^2 + 3y \)
- verifier checks:
  \( p_2(0) + p_2(1) = 0 + 6 = 6 \mod 13 \)
- verifier picks randomly: \( z_2 = 3 \)

Example
\[ p_3(3) = [(9 + 3) * \prod_{x \in \{1\}}(9z + 3(1-z)) + \prod_{w \in \{1\}}(z + 3(1-w))] \]

Round 3: (prover claims this = 7)
- everyone agrees expression = 12*(…)
- prover removes outermost \( \land \); sends
  \( p_3(z) = 8z + 6 \)
- verifier checks:
  \( p_3(0) * p_3(1) = 6(14) = 84; 12*84 = 7 \mod 13 \)
- verifier picks randomly: \( z_3 = 7 \)

Example
\[ 12*p_3(7) = 12 * [(9^7 + 3(1-7)) + \prod_{w \in \{1\}}(7 + 3(1-w))] \]

Round 4: (prover claims = 12*10)
- everyone agrees expression = 12*[6*(…)]
- prover removes outermost \( \Sigma \); sends
  \( p_4(w) = 10w + 10 \)
- verifier checks:
  \( p_4(0) + p_4(1) = 10 + 20 = 30; 12^[6+30] = 12*10 \mod 13 \)
- verifier picks randomly: \( z_4 = 2 \)
- Final check:
  \( 12^[9*7+(3(1-7))] = 12*[6+p_4(2)] = 12^[6+30] \)
Arthur-Merlin Games

- **IP** permits verifier to keep coin-flips private
  - necessary feature?
  - GNI protocol breaks without it

- Arthur-Merlin game: interactive protocol in which coin-flips are public
  - Arthur (verifier) may as well just send results of coin-flips and ask Merlin (prover) to perform any computation he would have done

**Theorem:** for all constant \( k \geq 2 \)
\[ \text{AM}^k = \text{AM}[2] \]

- **Proof:**
  - we show \( \text{MA}[2] \subset \text{AM}[2] \)
  - implies can move all of Arthur’s messages to beginning of interaction:
\[ \text{AMAMAM...AM} = \text{AAMAMAM...A} = \text{AAA...AMMM...M} \]

**MA and AM**

- Two important classes:
  - \( \text{MA} = \text{MA}[2] \)
  - \( \text{AM} = \text{AM}[2] \)

- definitions without reference to interaction:
  - \( L \in \text{MA} \iff \exists \text{poly-time language } R \)
  \[ x \in L \Rightarrow \exists m \Pr_{r}(x, m, r) \in R = 1 \]
\[ x \in L \Rightarrow \forall m \Pr_{r}(x, m, r) \leq \frac{1}{2} \]
  - \( L \in \text{AM} \iff \exists \text{poly-time language } R \)
  
  - set \( t = m+1 \) to get \( 2^{-t} < \frac{1}{2} \).
MA and AM

L ∈ AM iff ∃ poly-time language R

x ∈ L ⇒ Pr[∃m (x, m, r) ∈ R] = 1

x ∈ L ⇒ Pr[∃m (x, m, r) ∈ R] ≤ 1/2

• Relation to other complexity classes:
  - both contain NP (can elect to not use randomness)
  - both contained in \( \Pi_2 \): L ∈ \( \Sigma_2 \) iff R ∈ P for which:
    x ∈ L ⇒ Pr[∃m (x, m, r) ∈ R] = 1
    x ∈ L ⇒ Pr[∃m (x, m, r) ∈ R] < 1
  - so clear that AM ⊂ \( \Pi_2 \)
  - know that MA ⊂ AM

May 18, 2004 CS151 Lecture 15 25

MA and AM

• We know Arthur-Merlin = IP.
  - “public coins = private coins”

Theorem (GS): IP[k] ⊂ AM[O(k)]

– stronger result
– implies for all constant k ≥ 2,
  IP[k] = AM[O(k)] = AM[2]

• So, GNI ∈ IP[2] = AM

May 18, 2004 CS151 Lecture 15 27

Back to Graph Isomorphism

• The payoff:
  – not known if GI is NP-complete.
  – previous Theorems:
    if GI is NP-complete then PH = AM
    – unlikely!
  – Proof: GI NP-complete ⇒ GNI coNP-complete ⇒ coNP ⊂ AM ⇒ PH = AM

May 18, 2004 CS151 Lecture 15 29

New topic(s)

Optimization problems,
Approximation Algorithms,
and
Probabilistically Checkable Proofs
Optimization Problems

• many hard problems (especially \textbf{NP}-hard) are optimization problems
  – e.g. find shortest TSP tour
  – e.g. find smallest vertex cover
  – e.g. find largest clique

  – may be minimization or maximization problem
  – “opt” = value of optimal solution

Approximation Algorithms

• often happy with approximately optimal solution
  – warning: lots of heuristics
  – we want approximation algorithm with guaranteed approximation ratio of \( r \)
  – meaning: on every input \( x \), output is guaranteed to have value at most \( r \times \text{opt} \) for minimization
    at least \( \text{opt}/r \) for maximization

Approximation Algorithms

• Example approximation algorithm:
  – Recall:

    Vertex Cover (VC): given a graph \( G \), what is the smallest subset of vertices that touch every edge?

  – \textbf{NP}-complete

Approximation Algorithms

• Approximation algorithm for VC:
  – pick an edge \((x, y)\), add vertices \( x \) and \( y \) to VC
  – discard edges incident to \( x \) or \( y \); repeat.

  • Claim: approximation ratio is 2.
  • Proof:
    – an optimal VC must include at least one endpoint of each edge considered
    – therefore \( 2 \times \text{opt} \geq \text{actual} \)

Approximation Algorithms

• diverse array of ratios achievable
• some examples:
  – \((\text{min})\) Vertex Cover: 2
  – MAX-3-SAT (find assignment satisfying largest \# clauses): 8/7
  – \((\text{min})\) Set Cover: In \( n \)
  – \((\text{max})\) Clique: \( n/\log^2n \)
  – \((\text{max})\) Knapsack: \((1 + \varepsilon)\) for any \( \varepsilon > 0 \)

Approximation Algorithms

• called Polynomial Time Approximation Scheme (PTAS)
  – algorithm runs in poly time for every fixed \( \varepsilon > 0 \)
  – poor dependence on \( \varepsilon \) allowed

  • If all \textbf{NP} optimization problems had a PTAS, almost like \( P = \text{NP} \) (!)
Approximation Algorithms

• A job for complexity: How to explain failure to do better than ratios on previous slide?
  – just like: how to explain failure to find poly-time algorithm for SAT...
  – first guess: probably NP-hard
  – what is needed to show this?
• "gap-producing" reduction from NP-complete problem \( L_1 \) to \( L_2 \)

Approximation Algorithms

• “gap-producing” reduction from NP-complete problem \( L_1 \) to \( L_2 \)