Idea of Proof

Fix $x$, look at which choices of random bits make you accept
- By amplifying, can make probs very good
- Make copies by translating random bits with $\oplus$
- If $x \in L$, can cover all bits with a few copies, but can’t if $x \not\in L$

Reminders from Last Time

- **Goal:** $\text{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$
- Enough to show $\text{BPP} \subseteq \Sigma_2^p$ since $\text{BPP}=\text{co-BPP}$
- Obvious that $\text{RP} \subseteq \Sigma_2^p$ (and $\text{co-RP} \subseteq \Sigma_2^p$)
  - Issue here is two-sided error
    - PH can say “for all” or “exists”, but BPP says “for most”
- How do we fix this?
  - Picture on next slide
- **Notation:** $\oplus$ means bitwise XOR

More formally:
- Let $L \in \text{BPP}$ and $A(x,r)$ be poly TM s.t.
  - $A$ uses $m = \text{poly}(n)$ random bits
  - $\Pr[A(x,r) \text{ gives right answer}] \geq 1 - \frac{1}{3m}$
  - Using amplification
- **Claim:** $x \in L \iff \exists y_1, \ldots, y_m \in \{0,1\}^m \text{s.t.}$
  $$\forall z \in \{0,1\}^m \quad \sum_{i=1}^{m} A(x, y_i \oplus z)$$
Valiant-Vazirani Theorem

- What makes a SAT instance hard?
  - A lot of them are quite easy
  - Many people thought that having a lot of solutions made the problem *harder*
  - Why does this make sense?
- **Question**: Is SAT still hard if I promise you there’s \( \leq 1 \) satisfying assignment?
- **Answer**: Yes
- Important for a few reasons:
  - Refuted some conjectures at the time
  - Related to questions in cryptography
  - Same techniques will be useful for several other things

Proof Idea

- Say given SAT instance \( \phi(x) = \phi(x_1, \ldots, x_n) \)
- Will create (in rand. poly time) a new formula \( \psi(x) = \phi(x) \land p(x) \) s.t.
  - \( |\psi(x)| \) will be \( \text{poly}(|\phi|) \)
  - If \( \phi(x) \) is satisfiable, \( \psi(x) \) will have exactly 1 satisfying assignment with prob \( \geq 1/\text{poly}(n) \)
  - If \( \phi(x) \) unsatisfiable, \( \psi(x) \) will be unsatisfiable
  - This is automatic
- Say know that \( \phi \) has M solutions
- Choose rand. hash function \( h: \{0,1\}^n \rightarrow \{0,\ldots,4M\} \)
- Make \( p(x) \) be statement “\( h(x)=0 \)” 
  - Keeps each assignment with prob. \( 1/4M \)
  - Can show that this keeps exactly 1 with constant prob.
- Why doesn’t this work?
  - Don’t know M
    - Not that big a deal
  - Bigger problem:
    - If \( h \)=random hash function, description is exponentially large
    - So formula for \( p=^"h(x)=0" \) will be too big

Universal Hash Functions

- A family of functions \( H: \{0,1\}^n \rightarrow \{0,1\}^k \) is *universal* if
  \[
  \forall a_1 \neq a_2 \in \{0,1\}^n, \ b_1, b_2 \in \{0,1\}^k, \ 
  \Pr_{h \in \mathcal{H}} [h(a_1) = b_1] = \Pr_{h \in \mathcal{H}} [h(a_1) = b_1 \mid h(a_2) = b_2] = 2^{-k} \]
- Claim the following family is universal:
  - Choose matrix \( M \in \{0,1\}^{k \times n} \), vector \( y \in \{0,1\}^k \)
  - Let \( h_M(x) = Mx + y \pmod 2 \)
  - For any \( M,y \), can clearly write \( ^"h_M(x)=0^k" \) with poly sized formula
- Notes:
  - Need a family of functions, not just one
  - Pairwise independence vs. independence
  - Can be thought of as a partial derandomization
    - General hash function needs exp. random bits
    - Universal hash function only needs poly random bits