Today:

- SAT $\leq_{RP}$ Unique-SAT
- \#P, Counting Problems
- Permanent

Unique-SAT: Inspired by one-way permutations

$f: \{0,1\}^n \rightarrow \{0,1\}^n$ 1-1

"One way: given $x$ computing $f(x)$ is in P.

Given $y$ computing $x$ s.t. $f(x) = y$

is (hopefully) hard"

$NP = P \Rightarrow$ one-way permutations do not exist.

$\Leftrightarrow$ open.

Hurdle to proving $\Leftrightarrow$

Natural candidate: Given $y$ produce $\phi$ s.t. $\phi(x) = 1$

if $f(x) = y$.

But such a $\phi$ has unique sat. assignment!
Most NP-complete reductions will produce \( \Phi \) if \( \Phi \) has \( \text{exp(n)} \) many SAT assignments.

So challenge: Show even \( \Phi \)'s with unique SAT assign hard to satisfy.

Definition: How to define Unique SAT.

\[ \text{Unique SAT} = \{ \Phi \mid \Phi \text{ has exactly one SAT assignment} \} \]

Not right! Unique SAT \( \notin \text{NP} \) \( \text{at least not known} \).

Correct Defn: Promise Problem

\[ \text{USAT}_y = \{ \Phi \mid \Phi \text{ has exactly 1 SAT assignment} \} \]

\[ \text{USAT}_n = \{ \Phi \mid \Phi \text{ has no SAT assign} \} \]

Now USAT \( \leq \) SAT!
Can we show converse: $\text{SAT} \leq \text{USAT}$

- No known deterministic algorithm

  (Do not trust this statement... it's a vague recollection)

  "If $\text{SAT} \leq_k \text{USAT}$ (with Karp reduction) then
  hierarchy collapses?"

- [Valiant–Vazirani] $\text{SAT} \leq_{R^P} \text{USAT}$

---

**Randomized Reduction $B_P$ from problem $A = (A_y, A_N)$ to $B = (B_y, B_N)$**

- $A \leq_{B_P} B$ if $\exists$ randomized alg $f$ in prob. poly time
  - $x \in A_y \implies \Pr[f(x) \in B_y] \geq \frac{1}{2}
  - $x \in A_N \implies \Pr[f(x) \notin B_N] \leq \frac{1}{2}$

  [with $C - S \geq \frac{1}{\text{poly}(n)}$]

  If $S = 0$ then $A \leq_{R^P} B$. 
Proposition:

- \( A_{Bp} \leq_{Bp} B \leq_{Bp} \Rightarrow A \leq_{Bp} B \)
- \( A \leq_{RP} B \leq_{RP} \Rightarrow A \leq_{RP} B \)
- \( A \leq_{RP} B \leq_{RP} \Rightarrow ? \)

Theorem [VV]: \( SAT \leq_{RP} USAT \).

Proof idea:

- Will start with formula \( \phi \) & tach on \( \psi \) so that \( \# \) sat assignments reduce.
- Pick \( \psi \) at random so that \( \psi(x) \) satisfies.

- Clearly \( \psi \) w.p. \( 2^{-10} \).
- \( \phi \) & \( \psi \rightarrow \phi \land \psi \rightarrow \phi \land \psi \)
- \( \phi \) has \( 2^k \) sat \( \rightarrow \phi \land \psi \) has exactly 1 sat. assgmt.

Need to check idea - does it work?

But what is \( k \)?

How to "construct efficient \( \psi \)"?
Random $\Psi$ works if $\kappa$ is known and $\# \{x | \phi(x) = 1\} \in [2^{k-2}, 2^{k-1}]$.

Fix $x$ s.t. $\phi(x) = 1$.

$\Pr[\exists x \text{ s.t. } \phi(x) = 1] = 2^{-k}$

Fix $x' \neq x$ s.t. $\phi(x') = 1$.

$\Pr[\exists x' \text{ s.t. } \phi(x') = 1] = 2^{-k} (1 - 2^{-k})$.

$\Pr[\exists x' \text{ s.t. } \phi(x') = 1] \geq \frac{1}{10} \cdot 2^{-k}$.

$\Rightarrow \frac{1}{10} \cdot 2^{-k} \geq \frac{1}{40}$.

$\Rightarrow 2^{k-2} \cdot 1 \cdot 2^{-k} \geq \frac{1}{40}$.

1. $\# x \text{ s.t. } \phi(x) = 1$

2. $x$ uniquely satisfies $\Psi$

$\Rightarrow x'$ does not uniquely satisfy $\Psi$. 

reverse union bound for mutually exclusive events.
Efficient $\phi$?

Satisfies that $\Pr_{\psi} \left[ \psi(x) = 1 \land \psi(y) = 1 \right] = 4^{-k}$

$satisfies that \Pr_{\psi} \left[ \psi(x) = 1 \right] = 2^{-k}$

Pairwise independent hash families:

$X = \sum h : \{0,1\}^n \rightarrow \{0,1\}^k$ is p.w.i. if

$\forall x \neq y, \forall a, b \quad \Pr_{\psi} \left[ h(x) = a \land h(y) = b \right] = 4^{-k}$

$h \in X$

Example $\psi = \left( \left[ \begin{array}{c} M \end{array} \right], \left[ \begin{array}{c} b \end{array} \right] \right)$

$h(x) = M \cdot x + b$ [over $GF(2)$]

Exercise: Verify.

$satisfies that \Pr_{\psi} \left[ \phi \psi \text{ uniquely sat. by } x \right]$
Last hurdle: \( k \) unknown!

Solution: Guess \( k = n \pm 1 \) if \( k \in \mathbb{N} \)

Final Reduction

\[ \phi \xrightarrow{\text{Final Reduction}} (r, \phi \land \psi) \quad \text{where} \quad r \in \{n+1\}, \psi \in \mathcal{P} \]

Soundness: \( \phi \text{ UNSAT } \Rightarrow \phi \land \psi \text{ UNSAT}_n \) (w.p. 1)

Completeness: \( \phi \text{ SAT } \Rightarrow \phi \land \psi \in \text{USAT}_n \) (w.p. \( \frac{1}{40n} \))

Open: Amplify error probability!
Counting Problems

\[ f : \Sigma^* \rightarrow \mathbb{Z} \geq 0 \]

is a \#P function if there exists a \( \text{NP} \)-complete \( M(\cdot, \cdot) \) in \( \text{NP} \) such that

\[ \forall x \exists y | M(x, y) = 1 \]

\[ x \]

\[ \text{NP} \leq \#P \leq \text{PSPACE} \]

\[ \text{CoNP} \leq \#P \]

\[ \text{BPP} \]

Algebra \( \frac{\text{OR}}{\text{AND}} \rightarrow \frac{\text{SUM}}{\text{PRODUCT}} \)

\[ \text{Algebraic interest} \rightarrow \text{inherent interest} \]

\[ \exists x \forall y : C_j(x) \text{ satisfied} \Rightarrow \exists x \Pi \left( C_j(x) = 1 \right) \]

\[ \#P ! \]
# P - Complete Problems

- \# SAT
- \# Vertex Cover
- \# Hamiltonian Circuit

Karp reduction were "parsimonious"

\[ \phi \in SAT \Rightarrow \exists T(\phi) \in V.C. \]

(\( \phi, y \)) satisfied \( (T(\phi), T'(y)) \) satisfied for a known number of \( T' \)

- \# DNF (\# sat. assgnts of DNF formula)

\( \phi \rightarrow \overline{\phi} \)

\( N \) sat. assgnts \( \rightarrow \) \( 2^n - N \) sat.

- Network Reliability
- Partition function in Physics
- Bayesian Inference
Permanent

\[ M = \sum_{\pi \in S_n} M_{\pi(i)\pi(j)} \]

Combinatorial Interp.: \( \text{Perm}(M) = \# \text{ perfect matching in Bip. graph with adjacency matrix } M \)

Valiant: \( \#VC \leq \text{Perm} \)

Recall: \( VC \) = Subset of vertices that touch every edge.

Key Step 1: Gadgets for vertices + edges

Want:

- \( A, B, C, D \) chosen \( \Rightarrow \) 1 matching in Gadget
- \( A, B \) chosen \( \Rightarrow \) 1
- \( C, D \) chosen \( \Rightarrow \) 1
- none chosen \( \Rightarrow \) 0
Unfortunately, contradictory! No such gadgets exist if edges have positive weight.

Valiant’s Brilliant Idea: ① Use negative weight edges.

Permanent still well-defined but no comb. interpretation.

2nd Brilliant Idea:

Compute \( \text{Perm}(M) \mod p \).

Now all #’s positive 2 allows us to compute \( \text{Perm}(M) \) if \( p \) large enough, or by using many \( p^k \)’s.

Many nice prop. if \( p \) come from permanent.