Today's thm: $PH \subseteq P^{\#SAT}$.

Recall plan: 1. show $\forall c\ \exists f$ s.t. F Σ^c formula then
F true $\Rightarrow f(F) \in \oplus SAT$ w. pr $1 - 2^{-m}$
false $\Rightarrow$ 2^{-m}.

2. derandomize algo.

We saw base case of 1. for Σ^1 formulas.
Obs $\oplus P$ is closed under complementation (use $+1$).
Hence Π^1 formulas also OK.

Induction step. Assume F Σ^c formula:
$\exists \vec{x_1}\ G(x_1)$ where $G(x_1) \in \Pi^{c-1}$ and by induction can build $f(G(x_1))$.

Recall proof of VV thm:
F SAT $\Rightarrow f(F) \in \oplus$ SAT w.pr $\geq 1/8n$
F UNSAT $\Rightarrow$ $= 0$.
All we did was attach a hash function (encoded in α_F).

Therefore we can use the same idea and have

$\exists x_1$ s.t. $f(G(x_1)) \in \oplus SAT \Rightarrow f(G(x_1)) \wedge h(x_1) = 0 \in \oplus SAT$ w.pr $\geq 1/4n$
$\nexists x_1 \Rightarrow$ 0.

And also use the same error reduction technique to obtain a single formula

$H = W_1 \veebar W_2 \veebar \cdots \veebar W_s$ // recall special $\vee$: $F \veebar G = (F+1)\cdot(G+1) + 1$.

s.t. $f(G(x_1))$ SAT $\Rightarrow H \in \oplus SAT$ w.pr $\geq 1 - 2^{-m}$
UNSAT

obs we really wanted $G(x_1)$ SAT $\Rightarrow$ ---
but we can assume all $f(G(x_1))$ are equiv to $G(x_1)$ except w. prob. $s \cdot 2^{-m}$.
By taking small enough error from induction step we're good.

Step 1 ✓. Now do step 2: derandomize.

lemma: $\exists$ deterministic poly-time reduction (in $|F|, \ell$) $F \mapsto G$ s.t.
$F \in \oplus SAT \Rightarrow \#G \equiv -1 \pmod{2^\ell}$, $\ell = 2^i$.
$\notin \Rightarrow 0$.

proof: induction. $\ell = 1$ trivial by hyp.
o/w. consider $G = 4F^3 + 3F^4$.

say $x = \alpha + k \cdot 2^\ell$.

$4x^3 + 3x^4 = 4[\alpha^3 + 3\alpha^2 k 2^\ell + \_\_] + 3[\alpha^4 + 4\alpha^3 k 2^\ell + \_\_] =$
$= 4\alpha^3 + 3\alpha^4 + 12k \cdot 2^\ell [\alpha^2 + \alpha^3]$.

$\alpha = -1 \rightsquigarrow -1$.
$\alpha = 0 \rightsquigarrow 0$. ✓

proof of step 2.

Let f be rand. red from step 1, g red from lemma.

Consider $\sum_{r \in \{0,1\}^{\ell-1}} \#g(f(F,r)) \mod 2^\ell$.
(pick ℓ smallest power of 2 s.t. $|r| \leq \ell - 1$, pad r).

assume F true. then $g(f(1))$ is -1 w.pr. $\geq 1 - 2^{-m}$.
hence $\Sigma \in [-2^{\ell-1}, -2^{\ell-1} \cdot (1 - 2^{-m})]$
F false then 0 $\geq$.
hence $\Sigma \in [0, -2^{\ell-1} \cdot 2^{-m}]$.

hence we only need to distinguish $\Sigma \in$ ranges.
can we do that with a call to #SAT?

yes: write $g(f(F,r))$ as ckt, with inputs y, r.
convert into CNF $G(y, r, z)$, where z are aux vars uniquely def. by y, r, and s.t. $G(y,r,z)$ true iff y satisfies $g(f(F,r))$.

then $\#G = \sum_r \#g(\cdot)$ as we wanted.

algorithm: build G.
query oracle for $\#G$.
answer true iff answer is $\leq \Sigma$ mod 2^ℓ.

this completes proof of Toda's thm. ✱

Next we explore idea of derandomization further.

Derandomization

Goal: transform algorithms from BPP into P.
Obs if algo only uses polylog many random bits, then can make it deterministic.

We'll need pseudorandom generator that can "amplify" randomness from polylog to poly, give that to BPP algorithm, and make sure algo does not notice.

def Pseudorandom Generator. R distr. over $\{0,1\}^m$.
is (S, ε)- pseudor. if $\forall$ ckt C, $|C| \leq S$,
$\left| \Pr_{r \in R}[C(r) = 1] - \Pr_{r \in U_m}[C(r) = 1] \right| < \varepsilon$.

$G: \{0,1\}^* \to \{0,1\}^*$ exp-time computable is $S(n)$-PRG
if $|G(z)| = S(|z|)$ for every seed z and $\ell \in \mathbb{N}$
$G(U_\ell)$ is $(S(\ell)^3, 1/10)$- pseudor.

Let us see how to use PRG to derandomize.

lem. if a $S(n)$- PRG exists, then for every $\ell(n)$
$BPTIME(S(\ell(n))) \subseteq DTIME(2^{O(\ell(n))})$
// assuming S, ℓ poly-time computable; $\geq$ linear.

proof. $L \in BPTIME()$. $\exists$ TM M w/ error $\leq 1/3$

algo: for each seed $z \in \{0,1\}^{\ell(n)}$:
run $M(x, G(z))$.
answer majority.

claim: for large enough n, algo is correct.
assume not. then $\exists$ seq x_n s.t. $\Pr_z[M(x_n, G(z)) \neq L(x_n)] \geq 1/2$.
use to build ckt that distinguishes $G(U_{\ell(n)})$ and $U_{S(\ell(n))}$.

$C(r) = A(x_n, r)$ obs A runs in time $S(\ell)$,
hence $|C|$ is $\leq |S^2(\ell)|$
also must $n \leq S(\ell(n))$.
✱.