Rev Sess Mon. noon-2pm NSC 228 presume $t(n) \geq n+1$.

<u>Def^n</u>: A Turing machine M runs in time $t(n)$ if for all inputs $x \in \Sigma^*$, $M(x)$ halts within $n = |x|$ steps.

If M is an NTM, then we need <u>all</u> computations use $\leq t(n)$ steps

$DTIME[t(n)] = \{ L(M) : M$ is a DTM that runs in time $t(n) \}$

$NTIME(t(n)) = \{ L(N) : N$ is an NTM " " " " $\}$.

The class <u>P</u> ("Polynomial Time") is defined to be $\bigcup_{k=1}^{\infty} DTIME[n^k]$

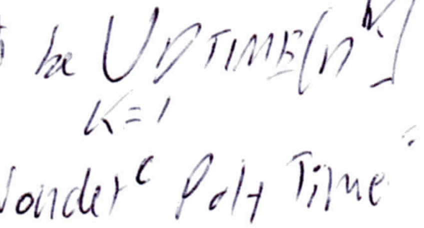
<u>NP</u> $=_{def} \bigcup_{k=1}^{\infty} NTIME[n^k]$. "Nondet Poly Time"

What belongs to P — which problems are decidable <u>in poly time</u>?

- Every regular language L is in P: Take a DFA M for L, and M "is" a DTM that runs in time $n+1$, in fact.

So $REG \subseteq P$

- More generally, the A_{DFA} problem is in P.



INST: A DFA M and an input w to M, coded as $x = \langle M, w \rangle$

QUESTION: Does M accept w?

Still backtracks inside M but takes $O(n_1 \cdot n_2) = poly(n)$ time

- How about A_{NFA}? INST: An NFA N, an $w \in \Sigma^*$ $\underbrace{[\ N \quad \# \ w\ ]}$ with n_1 over N and n_2 over w
  QUES: Does N accept w? [w]

w_1 w_2
$[\{s\} \ \{\ \} \ \ \{\ \} \ \{\ \} \ \ \{\ \} \ ]$

Not Kosher to convert N into a DFA M:
M can be $\approx 2^{n_1}$ times as big.
But we can trace the set of currently active states of N on input w directly.

or $E(\{s\})$ if there are ε-arcs out of s in N.

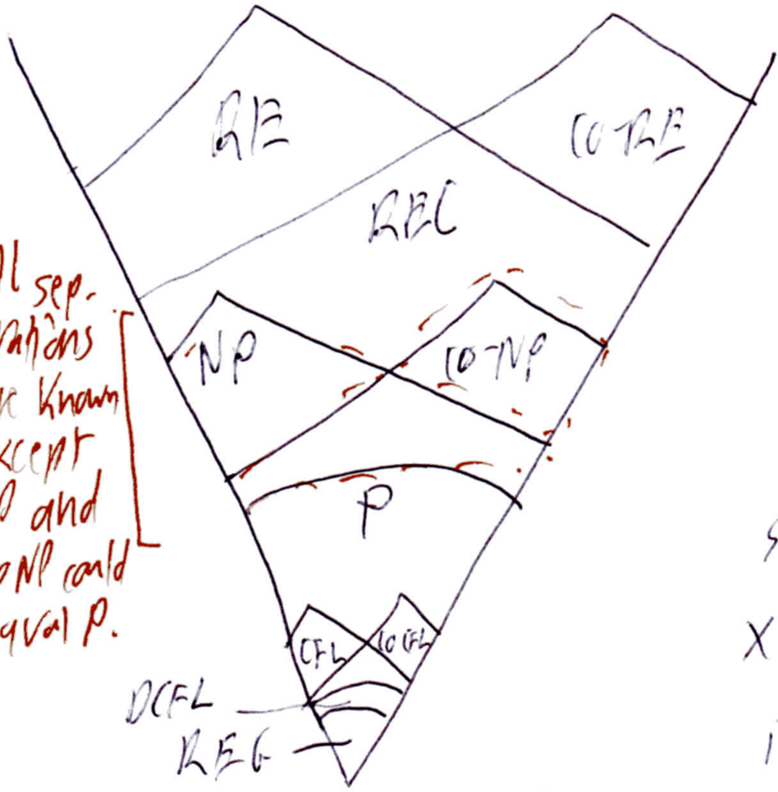
Each set we write down has size $\leq n_1$. It takes n_1 time to update each state in the set. So time $= O(n_1 \cdot n_1 \cdot n_2) =$ poly.

- How about E_{NFA}? Poly-time by BFS since it works on the graph of the NFA N. [NFA-to-DFA works on graph of M.]
- How about ALL_{NFA}? Well... converting NFA to DFA is "unkosher."
- FACT: Every CFL L belongs to P. The previous algorithm that took a CFG G in ChNF s.t. $L(G) = L$ and tried all derivations of length $2n-1$ does not run in poly(n) time given a $w \in \Sigma^n$.

In text (skim) There is a "Dynamic Programming" alg^m in $O(n^3)$ time.

Not In Text $A_{CFG} = \{\langle G, w\rangle : w \in L(G)\}$ is in P because there is a faster ChNF conversion not in the textbook.

- E_{CFG}, $EPS_{CFG} = \{\langle G\rangle : \epsilon \in L(G)\}$ are in P by "marking alg^{m}s."
  They use an unbounded while loop, but each iteration either marks a new variable or the whole thing stops: $\therefore$ time $O(|R| \cdot |V|)$
- How about ALL_{CFG}? Undecidable, so certainly not in P, nor NP.



All separations are known except NP and coNP could equal P.

Theorem (often used to <u>define</u> NP):

A language L belongs to NP if and only if there are a polynomial $q(n)$ and a language $V \in P$ {a polynomial-time verifying TM M_V} such that for all $x \in \Sigma^*$

$$x \in L \iff (\exists y : |y| \leq q(|x|)) \langle x, y \rangle \in V$$

ie. such that M_V accepts $\langle x, y \rangle$.

When $x \in L$, any such y is called a <u>certificate</u> or <u>witness</u>.

<u>Proof</u>: If $L = L(N)$ where N is an NTM running in time $t(n) = n^k$ then given $x \in L$, y can == an accepting config history of N on x. Time to verify $= O(\text{length of } y) \leq O(n^k \cdot n^k) = n^{2k} = \underset{q(n).}{\overset{|||}{poly(n)}}$.

Conversely, given q and M_V, build an NTM N that on any input x <u>guesses</u> y and <u>verifies</u> $\langle x, y \rangle \in V$ by running M_V. ⊠

<u>The Prime Example</u> of a language in NP <u>SATISFIABILITY</u> (SAT):

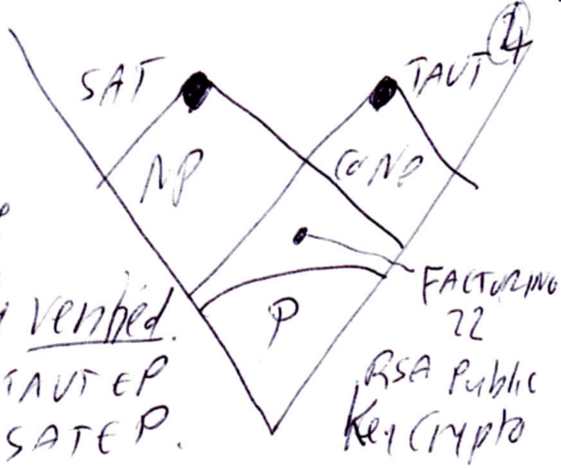
INST: A Boolean formula $\phi(x_1, \ldots, x_n)$ eg $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$

QUES: Is there a truth assignment $x_1 = a_1, \ldots, x_n = a_n$, ie $(a_1, \ldots, a_n) \in \{0,1\}^n$, that makes $\phi(a_1, \ldots, a_n) =$ <u>TRUE?</u>

ϕ can be satisfied, indeed by all assgts <u>except</u> $(0,1,0), (1,0,0)$.

Equivalently, is $\neg\phi$ <u>not</u> a tautology?

<u>TAUT</u> = {B.fs ψ : ψ <u>is</u> a tautology} is $\approx$ the complement of SAT.



Theorem: SAT $\in$ NP, also TAUT $\in$ coNP.

Proof: If ϕ is satisfiable, we can guess some $\vec{a} \in \{0,1\}^n$ st. $\phi(\vec{a}) = \text{true}$, which is easily verified. Since TAUT $\approx \overline{\text{SAT}}$, TAUT $\in$ coNP. ☒ So TAUT $\in$ P $\Leftrightarrow$ SAT $\in$ P.

Defn: A language A mapping reduces to a language B in polynomial time if there is a function f computable in poly time st. $\forall x: x \in A \Leftrightarrow f(x) \in B$. Diagram $A \nearrow B$ $>45°$ means $A \leq_m^p B$.

Theorem: If $B \in P \wedge A \leq_m^p B$, then $A \in P$; if $B \in NP$ then $A \in NP$.

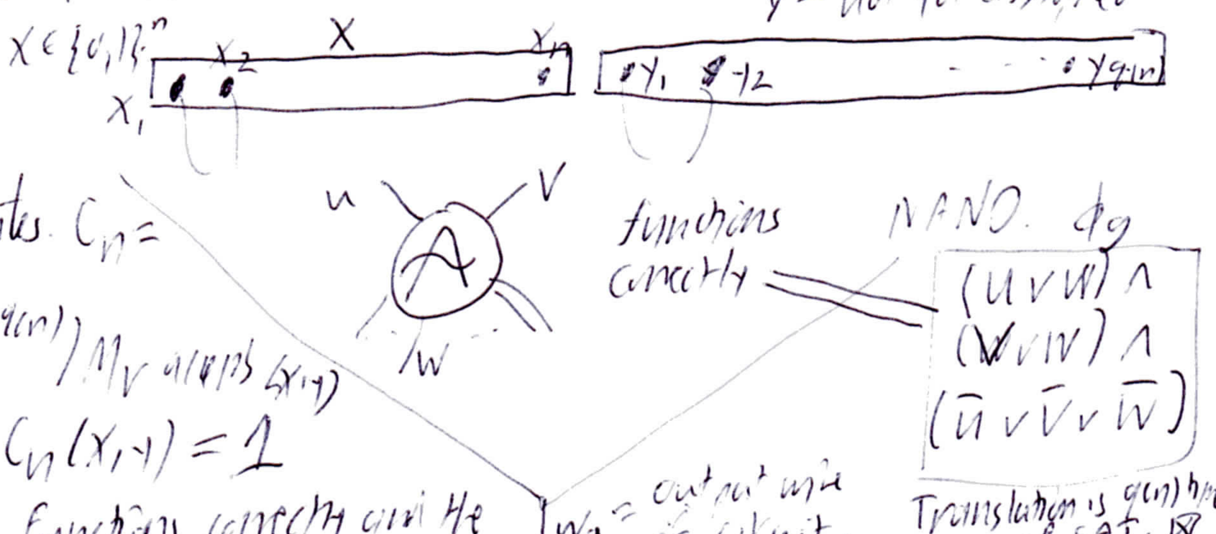
Defn: If $B \in NP$ and for all $A \in NP$, $A \leq_m^p B$, then B is NP-complete.

Theorem: SAT is NP-complete. The Cook-Levin Theorem, 1970-1

SAT $\in$ NP ✓

Proof: Let any $A \in NP$ be given. Show $A \leq_m^p$ SAT. Take $V \in P$ st. $\forall x: x \in A \Leftrightarrow \exists y \langle x, y\rangle \in V$ where $|y| \leq q(n)$ for a polynomial q.

"Burn" M_V into hardware as circuits of NAND gates. $C_n =$



$x \in A \Leftrightarrow (\exists y \in \{0,1\}^{q(n)})\, M_V$ accepts $\langle x, y\rangle$

$\Leftrightarrow (\exists y \in \{0,1\}^{q(n)})\, C_n(x, y) = 1$

$\Leftrightarrow$ Every NAND gate functions correctly and the output wire w_0 gives $1 \Leftrightarrow \phi \in \text{SAT}$, where $\phi = w_0 \wedge (\bigwedge \text{cls over all gates})$

Translation is $q(n)$ time so $A \leq_m^p$ SAT. ☒