TAUTOLOGY: (TAUT)

Inst: A Boolean formula ϕ (Phi) in variables $x_1, \ldots, x_n$ with AND, OR, NOT, possibly NAND connectives

Ques: Is ϕ a tautology, i.e. $\forall a \in \{0,1\}^n, \phi(a_1 \ldots a_n) = \text{True}$?

TAUT is decidable: test every row of the truth table for ϕ.

Problem is: truth table has 2^n rows. This doesn't scale: if the data size doubles, n vars $\hookrightarrow$ $2n$ vars, the time goes to

$2^{2n} = (2^n) \cdot 2^n$ Not a constant factor times the original time

Extra Defn: A running time $t(n)$ scales if there is a constant $C > 0$ s.t. $t(2n) \leq C \cdot t(n)$.

if $t(n) = n^2$ $\quad t(2n) = 4n^2 = 4 \cdot t(n)$

if $t(n) = n^3$ $\quad t(2n) = 8n^3 = 8 \cdot t(n)$

if $t(n) = n^k$ $\quad t(2n) = 2^k \cdot n^k = 2^k \cdot t(n)$.

If k is fixed, i.e. $t(n) =$ "polynomial", then we have "const scaling".

In fact, "$t(2n) \leq C \cdot t(n)$" $\Leftrightarrow$ "$t(n) =$ polynomial in n." (Assuming $t(n)$ is "smooth".)

Theorem: NP = P if and only if TAUT is in P.
i.e. iff tautology-solving algorithms can scale.

$\boxed{\begin{array}{l} \text{So: SAT} \equiv \overline{\text{TAUT}} \quad \text{(added)} \\ \text{SAT} = \{\phi : \neg\phi \notin \text{TAUT}\} \end{array}}$

Note: ϕ is not a tautology
$\iff$ there is an assignment $a \in \{0,1\}^n$ that makes $\phi(a_1 \cdots a_n) = $ False
$\iff$ there is an assgt $a \in \{0,1\}^n$ that makes $(\neg\phi)(a_1 \cdots a_n) = $ True
$\iff_{\text{defn}}$ there is a way to make $\neg\phi$ true $\equiv$ $\neg\phi$ is satisfiable.

Also note: if ϕ is a disjunction of terms $\phi = (x_1 \wedge \bar{x}_2) \vee (x_3 \wedge \bar{x}_1) \vee \cdots$

then $\neg\phi$ is a conjunction of clauses, called CNF $\neg\phi = (\bar{x}_1 \vee x_2) \wedge (\bar{x}_3 \vee x_1) \wedge \cdots$ $(x_3 \vee x_1)$
for Conjunctive normal form — (Not to be confused with Chomsky normal form, ChNF.)

SATISFIABILITY (General form, called SAT) We consider $n \approx |\phi|$.
INST: A Boolean formula $\phi(x_1 \cdots x_n)$
QUES: Is ϕ satisfiable, i.e. $(\exists a_1 \cdots a_n \in \{0,1\}^n) \cdot \phi(\vec{a}) = 1$?

CNF-SAT
INST: A Boolean formula ϕ in Conjunctive NF.
QUES: same (so this is a special case of SAT and trivially reduces to it like K reduces to A_{TM}.)

3SAT:
INST: A ϕ in CNF with at most (some say, exactly) 3 literals per clause.
QUES: same, so this is an even more special case.

Defn: A language B is NP-complete (under $\leq_m^p$) if • $B \in NP$ and
• for all $A \in NP$, $A \leq_m^p B$, meaning there is a function $f(x)$ computable in polynomial time s.t. $\forall x: x \in A \iff f(x) \in B$.

U Toronto 1970-71, Boston U. 1970-73

Theorem (Steve Cook and Leonid Levin) SAT, CNF-SAT, 3SAT are all NP-complete.

I SAT $\in$ NP Note that given an encoding $\langle \phi \rangle$ of ϕ:

$\langle \phi \rangle \in SAT \iff$ there exists $a_1, \dots a_n \in \{0,1\}^n$ st. $\underbrace{\phi(a_1 \dots a_n) = \text{True}}$

An NTM can guess $a_1 \dots a_n$ in n steps and then evaluate $\phi(a_1, \dots, a_n)$ in polynomial time. The text calls this latter stage a Verifier and uses the equivalent defn of NP:

A language B belongs to NP $\iff$ there is a polynomial $p(n)$ and a verifier V st for all x

$x \in B \iff (\exists y : |y| \le p(n))$ V accepts $x \# y$ within $p(|x|)$ steps.
$\langle x, y \rangle$

(7.20) Theorem: this is equivalent to $B = L(N)$ for some poly-time NTM N

$\Rightarrow$: Given V, build N to guess y and run $V(x \# y)$.

$\Leftarrow$ We can verify computations $\langle I_0, I_1, I_2, \dots, I_t \rangle$ for $t(n) \le p(n)$ because the language V_N from last lecture is in P and doesn't care whether the given computation is by a DTM or NTM. $\boxtimes$

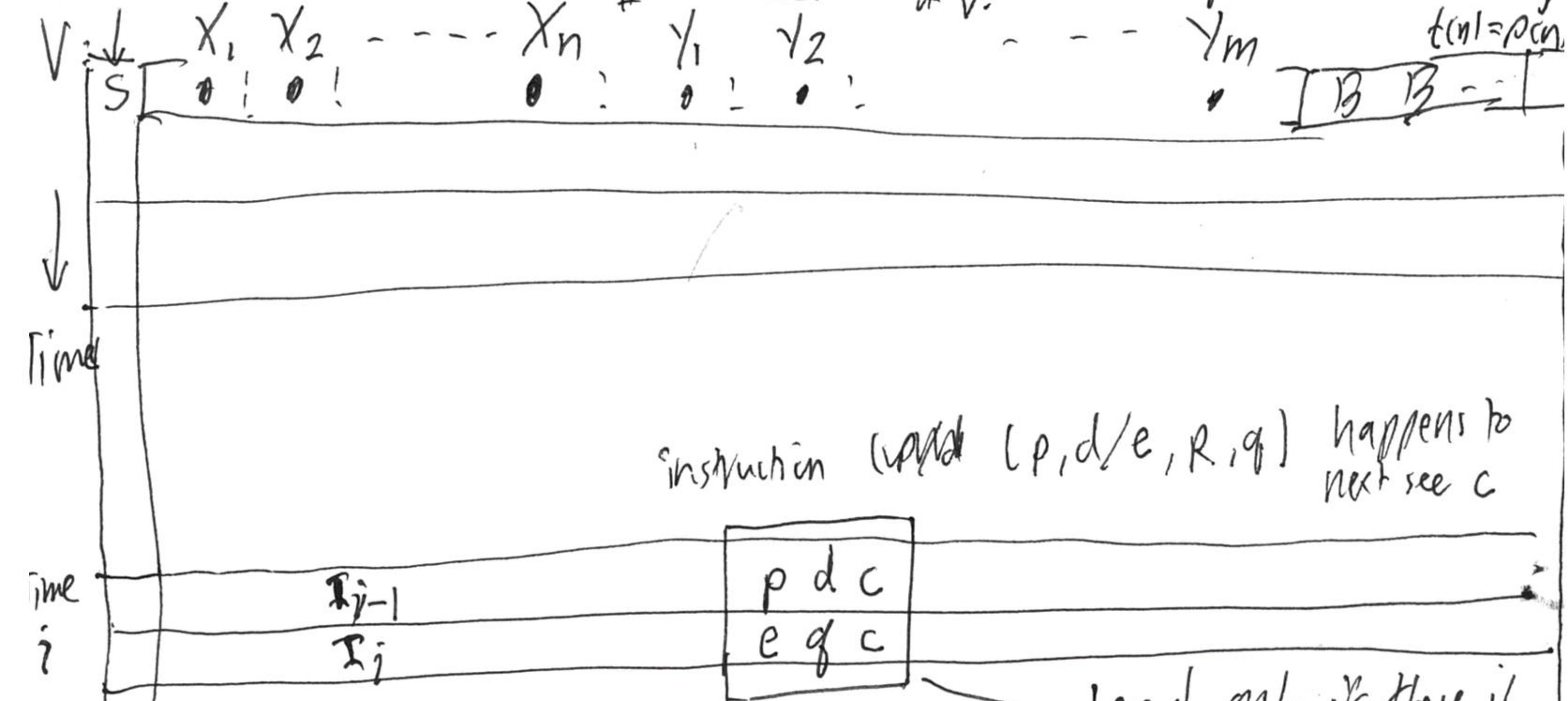
Since CNF-SAT and 3SAT are special cases they too belong to NP.

II. 3SAT is NP-hard: Let any $A \in NP$ be given. Goal: Show $A \le_m^p$ 3SAT

Take a polynomial time one-tape DTM V acting as verifier with runtime $p(n)$.

So $x \in A \iff \exists y_1 \dots y_{p(n)}$ V accepts $x_1 x_2 \dots x_n \# y_1 y_2 \dots y_m$ in $p(n)$

a binary word for $s, q \in Q$ etc.   # binary chars from original of V. $m = p(n)$, $n = |x|$ (4)



legal only if there is that instruction in δ_V. There is similar 2×3 box logic for a Left or Stationary move. This logic needs only a size-V gadget under binary encoding.

We can use the same 2×3 gadget overlaid everywhere in the circuit.

If the grid has no state, nothing happens — it's a no-op. We can burn this into a single Boolean circuit C_n of NAND gates

$t(n)$

q_{acc} q_{acc}

$t = p(n)$

W_0

$\phi_{g,w}$ $\equiv$

$(u \vee w) \wedge (v \vee w)$

$\wedge (\bar{u} \vee \bar{v} \vee \bar{w})$

Then $x \in A \Leftrightarrow (\exists y_1 \cdots y_m)\, C_n(x_1 \cdots x_n, y_1 \cdots y_m) = 1 \Leftrightarrow$ every NAND gate functions correctly and the output wire W_0 carries 1.

Our reduction function $f(x)$ outputs the 3CNF formula

$$\phi = (W_0) \wedge \bigwedge_{\substack{\text{gates } g \\ \text{wires } w \text{ out of } g}} \phi_{g,w}$$

then either substitute the actual values $b_1 \cdots b_n$ of the bits of x for the variables $x_1 \cdots x_n$, or use n more single clauses to force them, eg $b_1 = 1$, $b_2 = 0$ use $(x_1) \wedge (\bar{x_2})$.

ϕ has input variables $x_1 \cdots x_n$

guess variables $y_1 \cdots y_m$

wire variables $W_0, W_1, W_2 \cdots W_{O((p(n))^2)}$

Overall size of ϕ is $O(p(n) \times p(n))$. Thus N accepts $x \Leftrightarrow \phi \in 3SAT$.

So we have $f(x) = \phi$ based on C_n computable in $O(p(n)^2)$ = polynomial time, so

$$A \le_m^p 3SAT$$

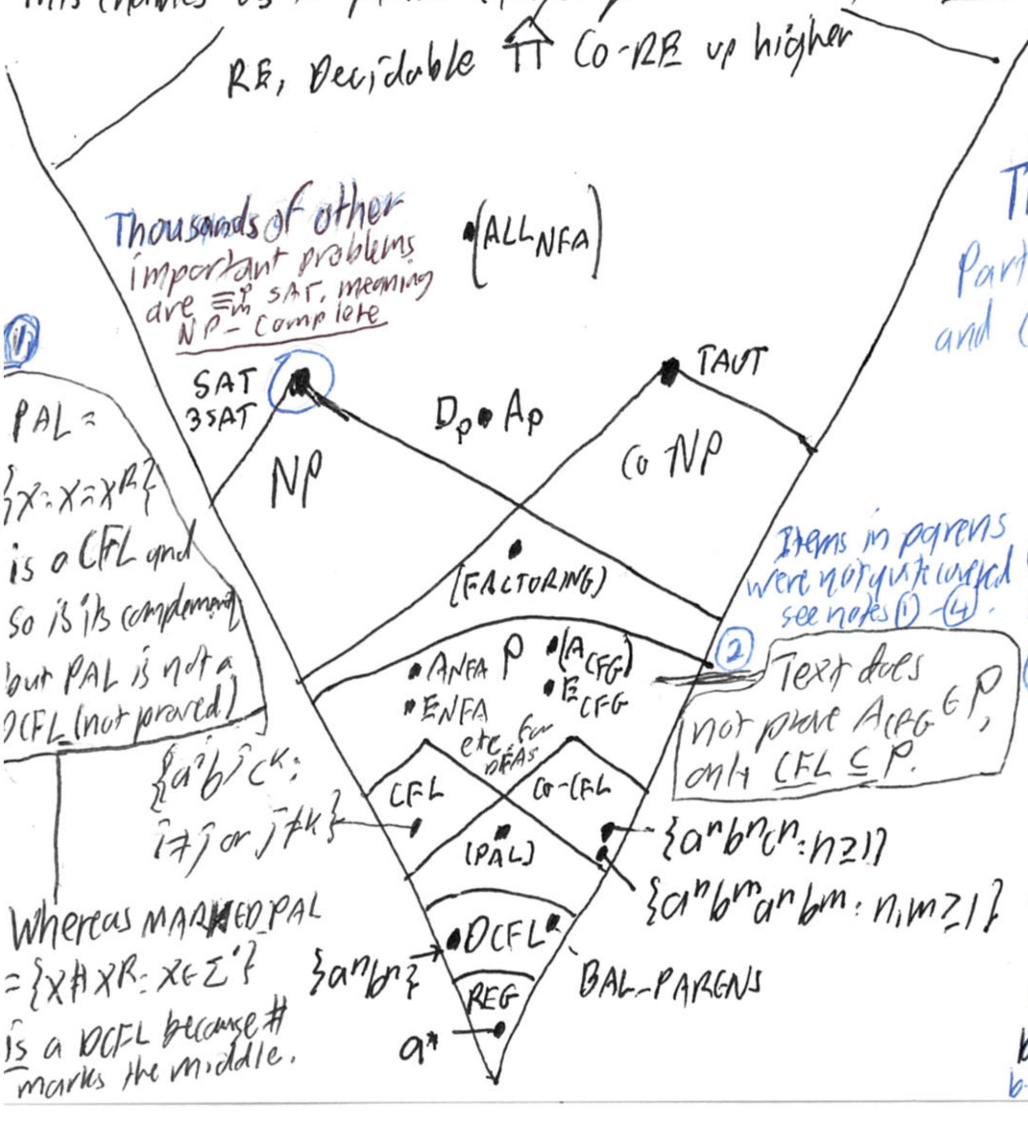
Since $A \in NP$ is arbitrary, 3SAT is NP-complete

Since $3SAT \le_m^p CNFSAT \le_m^p SAT$ "trivially", these versions are NP-complete too.

So $NP = P \iff SAT \in P \iff TAUT \in P$. ⊠

Added: This enables us to place languages into the final classes covered in the course:

The "Decidable" Lower Part of the Formal Languages and Complexity Landscape.



(4) ALL_{NFA} is NP-hard, and the problem in (3) of HW10 is NP-complete. Proof will be added to Key.

(3) As a language, FACTORING is $\{\langle x, w\rangle : w \le p$ for some prime factor p of $x\}$. call it FACT

Binary search on w can quickly find the biggest prime factor of x, so if FACT is in P then we can factor quickly and break codes like RSA. FACT is in $NP \cap$ co-NP because we can verify both 'yes' and 'no' by guessing the prime factorization of x