TAUTOLGY: (TAUT)  
\[ \text{Inst: A Boolean formula } \phi \text{ in variables } x_1, \ldots, x_n \]
\[ \text{with AND, OR, NOT, possibly NAND connectives} \]

\[ \text{Ques: Is } \phi \text{ a tautology, i.e. } \forall a \in \{0,1\}^n, \phi(a_n, \ldots, a_0) = \text{True} ? \]

TAUT is decidable: test every row of the truth table for \( \phi \).

Problem is: truth table has \( 2^n \) rows. This doesn't scale:
if the data size doubles, \( n \) rows \( \rightarrow \) \( 2n \) rows, the time goes to
\[ 2^n = (2^n) \cdot 2^n \] 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 \( t(n) = n^2 \)
  \[ t(2n) = 4n^2 = 4 \cdot t(n) \]
- If \( t(n) = n^3 \)
  \[ t(2n) = 8n^3 = 8 \cdot t(n) \]
- If \( t(n) = n^k \)
  \[ t(2n) = 2^k \cdot n^k = 2^k \cdot t(n) \]

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

In fact, \( t(2n) \leq c \cdot t(n) \) \( \Leftrightarrow \) \( t(n) \) = polynomial in \( n \).

Theorem: \( NP = P \) if and only if TAUT is in \( P \).

ie: fast tautology-solving algorithms can Scale.
Note: $\phi$ is not a tautology if:

- there is an assignement $a \in \{0,1\}^n$ that makes $\phi(a_1, a_n) = \text{False}$;
- there is an assignement $a \in \{0,1\}^n$ that makes $(\neg \phi)(a_1, a_n) = \text{True}$;
- there is a way to make $\neg \phi$ true $\equiv \neg \phi$ is satisfiable.

Also note: if $\phi$ is a disjunction of terms $\phi = (x_1 \land \overline{x_2}) \lor (x_3 \land \overline{x_1}) \lor \ldots$

then $\neg \phi$ is a conjunction of clauses, called CNF.

for Conjunctive Normal Form — (Not to be confused with Chomsky normal form, Ch NF.)

Satisfiability (general form, called SAT) We consider:

- INST: A Boolean formula $\phi(x_1, \ldots, x_n) \land \neg[\phi]$;
- QUES: Is $\phi$ satisfiable, i.e. $(\exists a_1, \ldots, a_n \in \{0,1\}^n) \cdot \phi(a) = 1$?

CNF-SAT

- INST: A Boolean formula $\phi$ in Conjunctive NF.
- QUES: same. (So this is a special case of SAT and trivially reduces to it like $\lor$ reduces to $\And$.)

3SAT:

- INST: A $\phi$ in CNF with at most 3 literals per clause.
- QUES: same, so this is an even more special case.

Defn: A language $B$ is NP-complete (under $\leq_p$) if $B \in \text{NP}$ and

- for all $A \in \text{NP}$, $A \leq_p B$, meaning there is a function $f(x)$ computable in polynomial time $\forall x \cdot x \in A \iff f(x) \in B$. 
Theorem (Steve Cook and Leonid Levin) SAT, CNF-SAT, 3SAT are all NP-complete.

I. SAT ∈ NP

Note that given an encoding \( \langle \phi \rangle \) of \( \phi \):

\[ \langle \phi \rangle \in \text{SAT} \iff \text{there exists } a_1,..,a_n \in \{0,1\}^n \text{ s.t. } \phi(a_1,..,a_n) = 1 \]

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

A language \( B \) belongs to NP \iff there is a polynomial \( p(n) \) and a verifier \( V \) s.t. for all \( x \in B \):

\[ x \in B \iff (\exists y : 1 \leq |y| \leq p(n)) \ V \text{ accepts } x \# y \text{ within } p(|x|) \text{ steps}. \]

\( \# \text{-NP} \)

Theorem. This is equivalent to \( B \subseteq \text{L}(N) \) for some polynomial \( N \)

\[ \Rightarrow \quad \text{Given } V, \text{ build } N \text{ to guess } x \text{ and run } V(x \# y). \]

\[ \Leftarrow \quad \text{We can verify computations } \langle I_0, I_1, I_2, ..., I_t \rangle \text{ for } t \leq p(n) \]

because the language \( V \) from last lecture is in \( P \) and doesn't care whether the given computation is by a DTM or NTM.

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

II. 3SAT is NP-hard: Let any \( A \in \text{NP} \) be given. Goal: show \( A \leq_m \text{3SAT} \)

Take a polynomial time \( \text{DTM} V \) acting as verifier with runtime \( p(n) \).

So \( x \in A \iff \exists y_1 ... y_m \text{ s.t. } V \text{ accepts } x \# y_1 ... y_m \text{ in } p(n) \text{ steps} \).
Our reduction function $f(x) = \phi(x)$ outputs the 3CNF formula $\phi$ was input variables $x_1, x_2, \ldots, x_n$.

Then $x \epsilon A \iff \exists y \epsilon B \forall z \epsilon C \exists w \epsilon D (x \lor y \lor z \lor w)$. Thus $N$ variables $x = \phi \epsilon \{0, 1\}$.

Next either satisfy each variable $x_i, x'_i$ or use in which clause.

Then every variable $y, y'$ are handled.

Finally, output the logical form of $x \lor y \lor z \lor w$.

We can use the same $3 \times 3$ gadget overlap everywhere in the circuit. If the grid you are using a new gadget, we can draw this in a single $A \lor (B \lor C \lor D)$. If we overlap, we can see the logic of $A \lor B \lor C \lor D$ from the gadget. If the logic only if then $A \lor B \lor C \lor D$ happens to be the gadget in $A \lor B \lor C \lor D$. This logic needs only a single $\lor$ gadget under binary encoding.
So we have \( f(x) = \phi \) based on \( C_n \) computable in \( O(p^c n^2) = \text{polynomial time} \), so

\[ A \leq^p \text{3SAT} \]

Since \( A \in \text{NP} \) is arbitrary, \text{3SAT} is \text{NP-complete}.

Since \( \text{3SAT} \leq^p \text{CNFSAT} \leq^p \text{SAT trivially} \), these reductions are \text{NP-complete}.

So \( \text{NP} = \text{P} \Leftrightarrow \text{SAT} \in \text{P} \Leftrightarrow \text{TAUT} \in \text{P} \).

Added:
This enables us to place languages into the final classes covered in the course:
- \( \text{RE} \), Decidable \( \iff \) \text{Co-1CFL} or higher

The "Decidable" lower part of the Formal Languages and Complexity Landscape.