(Cook's Theorem):

- **Theorem 9.1:** SAT is NP-Complete.

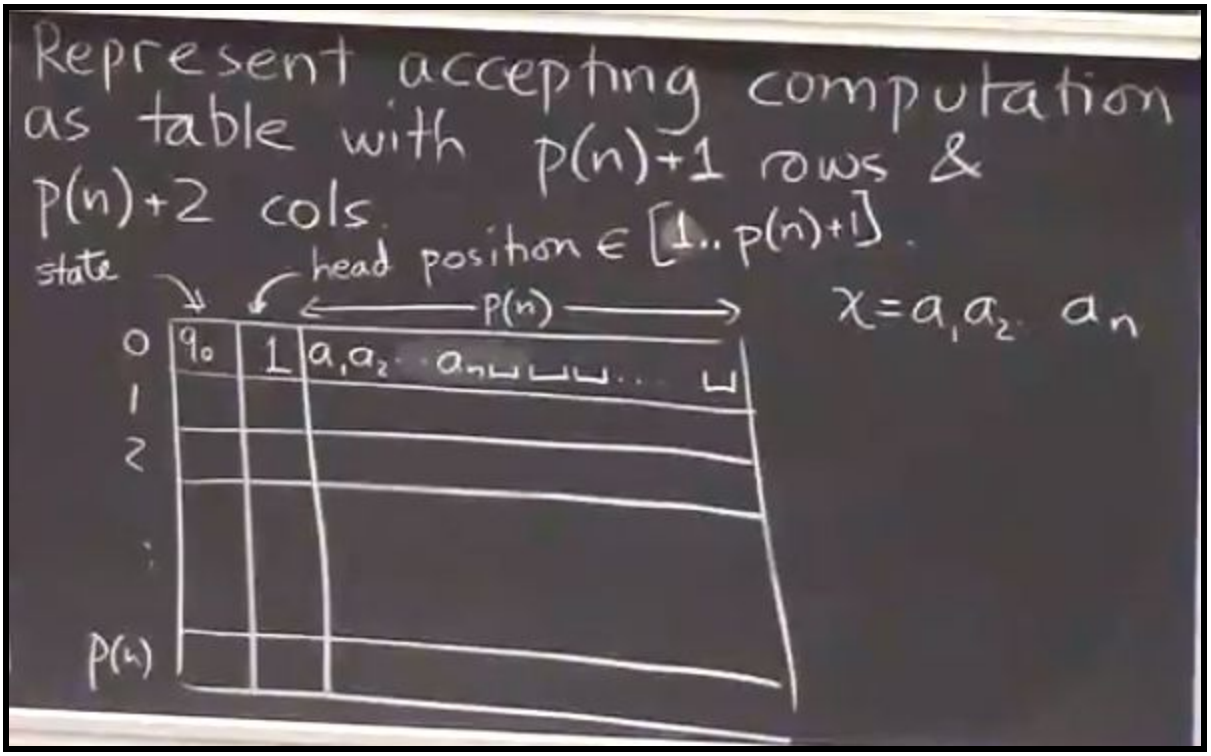
Proof:

1. **Proof that SAT ∈ NPC:**
   Note: This proof was done in lecture 8.
   The certificate in this case is a truth assignment that makes the formula true. The verifier evaluates the formula given the certificate.
2. **Proof that ∀ L ∈ NP, L ≤$_P$ SAT:**
   This part is tough because there is an infinite number of languages in NP.
   We can't prove each language ≤$_P$ SAT.
   Instead, we need to find something that all these languages have in common to help prove the reduction.
   We will call this item that all the languages have in common "a handle".
   For every language, L, in NP, the handle will be a NTM, M_L, that accepts L.
   Since the languages are in NP, we know that there exists a NTM that decides it in polynomial time.

   Given M_L and an input string x, where x ∈ ∑*, we will construct, in polytime, a function F_x s.t. x ∈ L iff F_x is satisfiable.
   We can rephrase the above statement like this: F_x is satisfiable means that there exists a truth assignment τ that satisfies F_x iff M_L on x has an accepting computation.

   Let p(n) be a polynomial bound on the running time of M_L, where n = |x|.
   This means that every computation of M_L on x takes at most p(n) steps, where n = |x|.
   M_L on x has an accepting configuration means that there exists a sequence of configurations C0⊢C1⊢…⊢Cl where C0 is the initial configuration, C0 = Q0x, and Cl is an accepting configuration.
   WLOG assume that p(n) ≥ n. (This means that the number of steps for the accepting computation is at least as long as the length of the input.)
   If p(n) < n, then we can just add useless steps to get p(n) ≥ n.
   If it was polynomial before, it's still polynomial.
   The tape on each C_t has at most p(n) symbols before the infinite number of blank symbols.
   Instead of thinking C0⊢C1⊢…⊢Cl as configurations, we will think of it as a table.
   There will be p(n)+1 rows, one row for each C_t.
   If l < p(n), I will still force the table to have p(n) + 1 rows. I will copy the last configuration to fill the remaining rows.
   Furthermore, there will be p(n)+2 columns. These columns will be used to represent the elements that are part of the configuration.
   We will put the state in the first column.
   We will put a number that represents the position of the head in the second column.
   We will put the p(n) symbols in the remaining p(n) columns, with one symbol per column.

Here's a diagram of the table:



The i^{th} row in the table is just another way to represent C_i.
We are changing the representation to make it more uniform.
ML = (Q, Σ, Γ, δ, Q0, QA, QR)
∀ q∈Q and ∀ t∈[0…p(n)], I will have a variable S_t^q, s.t. S_t^q = 1 iff at time t, the M_L is in state q. S_t^q = 0 otherwise.
∀ cell i∈[1…p(n)] and ∀ a∈Γ and ∀ t∈[0…p(n)], I will have a variable C_t^{ia}, s.t. C_t^{ia} = 1 iff at time t, cell i contains symbol a. C_t^{ia} = 0 otherwise.
∀ i∈[1…p(n)] and ∀ t∈[0…p(n)], I will have a variable H_t^i, s.t. H_t^i = 1 iff at time t, the head of the tape is on cell i. H_t^i = 0 otherwise.
The total number of variables is $\Theta(p^2(n))$.
We will use groups of formulas to help create F_x.
Group 1 (Coherence): At any time t, I cannot have:

a. M_L be in 2 different states.
b. M_L's head cannot be in 2 different cells.
c. Each cell cannot have 2 symbols.

Group 2 (Start Well): At time 0, M_L starts correctly.
Group 3 (End Well): At time p(n), M_L is in the accept state.
Group 4 (Move Well): In each step from t to t + 1:

a. Only the symbol under the head can change.
b. The state, head position, tape contents change as per δ.

For Coherence:

a. $\neg(S_t^q \wedge S_t^p)\ \forall p \neq q \in Q,\ \forall t \in [0...p(n)]$
   Note: We can express the above as: $\neg S_t^q \vee \neg S_t^p$
b. $\neg(H_t^i \wedge H_t^{i'})\ \forall i \neq i' \in [1...p(n)+1],\ \forall t \in [0...p(n)]$
   Note: We can express the above as: $\neg H_t^i \vee \neg H_t^i$
c. $\neg(C_t^{ia} \wedge C_t^{ib})\ \forall a \neq b \in \Gamma,\ \forall t \in [0...p(n)]$
   Note: We can express the above as: $\neg C_t^{ia} \vee \neg C_t^{ib}$

For Start Well:

M_L starts correctly means that at time 0,

a. The state is Q0. (S_0^{Q0})
b. The head is in cell 1. (H_0^1)
c. The tape is unchanged.
   ($C_0^{iai}\ \forall i \in [1...n]$) ← This means at time 0, cell i contains a_i for all i in 1 to n
   ($C_0^{i\_}\ \forall i \in [n+1...p(n)]$) ← This means at time 0, cell i contains a blank symbol for all i in n+1 to p(n).

For End Well:

We can represent "At time p(n), M_L is in the accept state." with $S_{p(n)}^{QA}$.

For Move Well:

a. $(C_t^{ia} \wedge \neg C_{t+1}^{ia}) \rightarrow H_t^i\ \forall i \in [1...p(n)+1],\ \forall a \in \Gamma,\ \forall t \in [0...p(n)]$
   Note: We can express the above as: $\neg C_t^{ia} \vee C_{t+1}^{ia} \vee H_t^i$.
b. $(S_t^q \wedge H_t^i \wedge C_t^{ia}) \rightarrow \vee(S_{t+1}^p \wedge H_{t+1}^{i+d} \wedge C_{t+1}^{ib})\ (p, b, R) \in \delta(q, a)$
   Note: We can express the above as:
   $\neg S_t^q \vee \neg H_t^i \vee \neg C_t^{ia} \vee (\vee(S_{t+1}^p \wedge H_{t+1}^{i+d} \wedge C_{t+1}^{ib})\ (p, b, R) \in \delta(q, a))$

$$d = \begin{cases} 1 & \text{if } D = R \\ -1 & \text{if } D = L \text{ and } i \neq 1 \\ 0 & \text{if } D = L \text{ and } i = 1 \end{cases}$$

I.e.
d = 1 if we move right.
d = -1 if we move to the left and are not in the leftmost cell.
d = 0 if we move to the left and are already in the leftmost cell.
$\forall p, q \in Q$ s.t. $q \neq QA,\ \forall i \in [1..p(n)+1],\ \forall t \in [0...p(n)]$
If q is in an accept state, there is no transition outside of an accept state.

If we are in QA, and there's empty rows in the table, we get
$(S_t^{QA} \wedge H_t^i \wedge C_t^{ia}) \rightarrow (S_{t+1}^{QA} \wedge H_{t+1}^i \wedge C_{t+1}^{ia})\ \forall i \in [1..p(n)+1],\ \forall t \in [0...p(n)]$, and $\forall a \in \Gamma$ or equivalently $\neg S_t^{QA} \vee \neg H_t^i \vee \neg C_t^{ia}) \vee (S_{t+1}^{QA} \wedge H_{t+1}^i \wedge C_{t+1}^{ia})$
Basically, we just copy everything from the row where we reach QA to all the empty rows.

Let F_x^1, F_x^2, F_x^3, F_x^4 be formulas in groups 1-4.
$F_x = F_x^1 \wedge F_x^2 \wedge F_x^3 \wedge F_x^4$
Claim: M_L accepts x iff F_x is satisfiable.
Proof:
(=>)
Suppose M_L accepts x.
Then, there exists a sequence of configurations C0, C1, …, Cl s.t.

- C0 is the initial configuration, C0 = Q0x
- $Ct \vdash Ct+1 \ \forall \ t \in [0..l-1]$
- Cl is the accepting configuration, Cl = Qa
- $l \leq p(n)$.

The contents of the table suggest a truth assignment that satisfies F_x.

(<=)
Suppose F_X is satisfiable.
That means that there exists a truth assignment that makes F_X true.
This truth assignment defines the rows of the table that correspond to an accepting computation.
Each row corresponds to a configuration of M_L. (Group 1)
The first configuration is the initial configuration of M_L on x. (Group 2)
The last configuration is an accepting configuration. (Group 3)
Each configuration follows from a previous by a legal move of M_L. (Group 4)

The reduction is polytime because the length of F_x is polynomial w.r.t |x| = n
Group 1's Formula is $O(p^3(n))$.
Group 2's Formula is $O(p(n))$.
Group 3's Formula is $O(1)$.
Group 4's Formula is $O(p^2(n))$.
Furthermore, each formula has size O(1).
$|F_x| = O(p^3(n))$ is polynomial w.r.t to |x|.

Examples of Other NPC Problems:

- We're going to prove that other problems, X, are NPC by showing $SAT \leq_p X$.
- However, before we do this we need to show that SAT is still NP C even if we restrict some formulas in a syntactic sense.
- **Definition:** F is in **Conjunctive Normal Form (CNF)** iff:
    a. It is a **literal**. A literal is a variable or the negation of a variable. OR
    b. It is a conjunction of **clauses**. A clause is a literal or a **disjunction** of literals.

  I.e. CNF is an $\wedge$ of $\vee$s, where $\vee$ is over variables or their negations (literals). An $\vee$ of literals is also called a clause.

  E.g. The following formulas are in CNF:
  $\neg X \rightarrow$ Is a literal
  $\neg X \wedge \neg Y \rightarrow$ Is a clause because it is a disjunction of 2 literals.
  $\neg x_1 \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3) \rightarrow$ Is a clause because there are disjunctions of literals.

  E.g. The following formulas are not in CNF:

¬(X ∨ Y) since an OR is nested within a NOT.

- **Note:** Every formula can be equivalently written as a formula in CNF.
- **CNF-SAT:**
- **Theorem 9.2:** CNF−SAT ∈ NPC.

Proof:

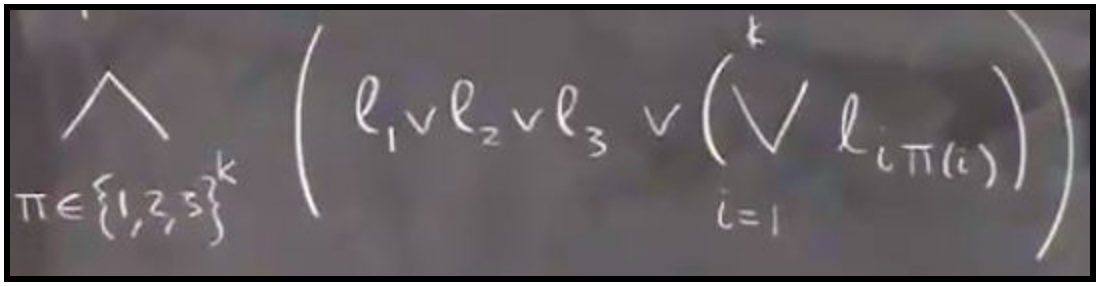
F_x is almost CNF.

The exception is in group 4.

We have $l_1 \vee l_2 \vee l_3 \vee (l_{11} \wedge l_{12} \wedge l_{13})) \vee ... \vee (l_{k1} \wedge l_{k2} \wedge l_{k3}))$ where l represent literals.

This is in DNF not CNF.

However, using distributive laws this is logically equivalent to



The size of this formula is $(3^k)(k+3)$ where $k \leq |Q||\Gamma|2 = O(1)$.

So, $(3^k)(k+3)$ is still a constant. $(3^k)(k+3) = O(1)$.

Put F_x in CNF as above.

The resulting formula has size polynomial w.r.t |x|.

Therefore, CNF−SAT ∈ NPC

- **Problem (3SAT):** CNF−SAT for formulas where each clause has ≤ 3 literals.
  Theorem 9.3: 3SAT ∈ NPC

Proof:

a. **Proof that 3SAT ∈ NP:**
   3SAT is a specific case of SAT which is in NP.
   Hence, 3SAT is in NP.
b. **CNF−SAT $\leq_p$ 3SAT:**
   Given any CNF formula F = C1 ∧ C2 ∧ ... ∧ Ck where Ci is a clause, construct in polytime a 3CNF formula F' = C'1 ∧ ... ∧ C'k.
   F' is satisfiable iff F is satisfiable.
   C'j is a 3CNF formula.
   |C'j| = O(|Cj|)
   If Cj has at most 3 literals, C'j = Cj.
   If Cj = l1 ∨ l2 ∨ … ∨ lm, where m ≥ 3, we introduce new variables z1, ..., zm−3.
   Then, let C'j = (l1 ∨ l2 ∨ z1) ∧ (¬z1 ∨ l3 ∨ z2) ∧ (¬z2 ∨ l4 ∨ z3) ∧ ... ∧ (¬zm−3 ∨ lm−1 ∨ lm).
   We are using Zi to chain the li.
   C'j is satisfiable iff Cj is satisfiable.
   Hence, F' is satisfiable iff F is satisfiable.
   |F'| = O(|F|) → CNF−SAT $\leq_p$ 3SAT

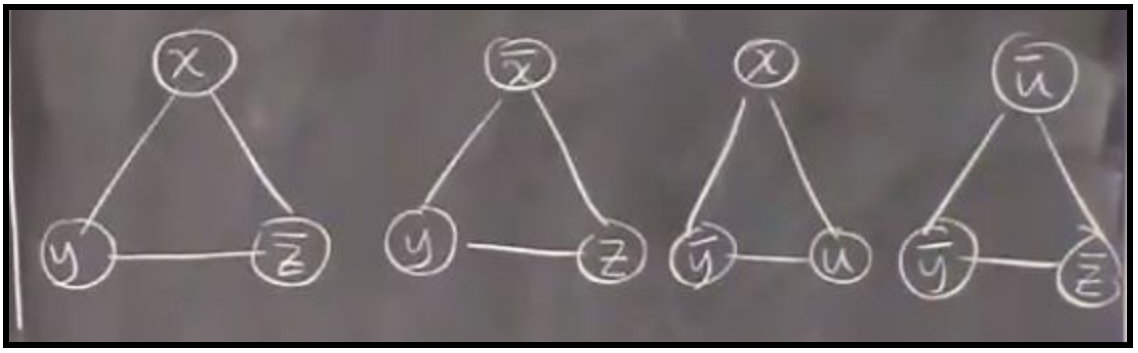
- **Fact:** 2SAT ∈ P

- **Problem (IS):**
  Instance: $\langle G, k \rangle$ where $G = (V, E)$ is an undirected graph and $k \in Z^+$.
  Question: Does G have an independent set $V' \subseteq V$ such that $|V'| \geq k$?

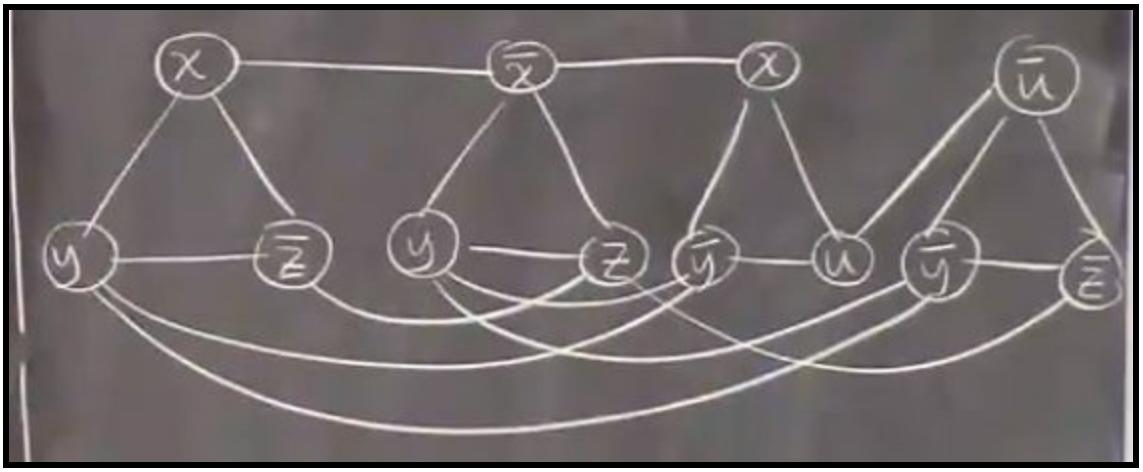
  Theorem 9.4: IS $\in$ NPC

  Proof:
  a. **Proof that IS $\in$ NP:**
     We have done this in lecture 8.
  b. **Proof that 3SAT $\leq_P$ IS:**
     Given 3CNF formula F, construct (in polynomial time in $|\langle F \rangle|$), an undirected graph G = (V, E) and $k \in Z^+$ s.t. F is satisfiable iff G has an IS of k nodes.
     Let $F = C1 \wedge ... \wedge Cm$.
     Let X1, …, Xm be variables in F.
     $Cj = (l_j^1, l_j^2, l_j^3)$ where l_j^t is a literal.

     E.g.
     Suppose we have the formula $F = (x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (x \vee \neg y \vee u) \wedge (\neg u \vee \neg y \vee \neg z)$.
     For each clause construct a triangle.



     Selecting a node is equivalent to saying that the truth assignment for that literal is true.
     E.g. In the first triangle, if we choose $\neg z$, we're saying that $\neg z$ is true, meaning that z is false.
     We must restrict not being able to choose opposing literals (x and $\neg x$) by drawing an edge between them.
     This is because by choosing x in 1 triangle and $\neg x$ in another triangle, we're saying that we want x to be true in the first triangle and $\neg x$ to be true in the second triangle. This will cause a contradiction.
     This is shown in the picture below.



k = 4 is the number of clauses we have.

In general,
G = (V,E) where

V = $\{C_j^1, C_j^2, C_j^3 \mid 1 \le j \le m\}$

E = $\{\{\{C_j^1, C_j^2\}, \{C_j^2, C_j^3\}, \{C_j^3, C_j^1\} \mid 1 \le j \le m\} \cup$
$\{\{C_j^t, C_{j'}^{t'}\} \mid 1 \le j \ne j' \le m,\ 1 \le t, t' \le 3\}\}$ s.t. l_j^t and $l_{j'}^{t'}$ are opposite literals of the same variable.

k = m

Claim: F is satisfiable iff G has IS of size k = m.

Proof:
(=>)
Suppose F is satisfiable.
Let τ be a truth assignment that satisfies F.
τ makes true some literal, say l_j^{tj} of every clause Cj $1 \le j \le m$, $1 \le tj \le 3$.
Then $\{C_1^{t1}, C_2^{t2}, \ldots C_m^{tm}\}$ is an IS of G of size m = k.
They belong to different triangles and there are no edges between them.
If there is an edge between 2 of them, say C_i^{ti} and C_j^{tj}, then they are opposite literals of the same variable. Hence, τ would make one of them true and the other false. However, we are only choosing the literal that τ makes true, so we can't choose both C_i^{ti} and C_j^{tj}. Hence, there cannot be any edges between any 2 (C_x^{tx}, C_y^{ty}) in $\{C_1^{t1}, C_2^{t2}, \ldots C_m^{tm}\}$.

(<=)
Suppose G has an IS, V', of size m.
Because of the triangles, V' has exactly one node from each triangle.
Let V' = $\{C_1^{t1}, C_2^{t2}, ..., C_m^{tm}\}$.
We will define truth assignment τ as:

$$\tau(x) = \begin{cases} 1, & \text{if } \exists_j \text{ such that } \ell_j^{t_j} = x \\ 0, & \text{if } \exists_j \text{ such that } \ell_j^{t_j} = \overline{x} \\ 0/1 & \text{otherwise} \end{cases}$$

If l_j^{tj} = x, then we make x = 1 (x = true).
If l_j^{tj} = ¬x, then we make x = 0 (x = false), so that ¬x is true.
This is well-defined, meaning that there are no contradictory rules, that satisfies F because it solves every clause.

Remains to show that construction of ⟨G,k⟩ can be done in polynomial time in size of ⟨F⟩.
|V | = 3m
|E| = 3m + 3m(m − 1) = $O(m^2)$.
Therefore, the size of ⟨G,k⟩ is polynomial w.r.t |⟨F⟩|.
Therefore, ⟨G,k⟩ can be constructed from ⟨F⟩ is polynomial.
Therefore, 3SAT $\leq_P$ IS.
Therefore, 3SAT ∈ NPC.

- **Problem (CLIQUE):**
  Instance: ⟨G,k⟩ as in IS.
  Question: Does G have a clique of size k?
  A **clique** is a subset of nodes, V', s.t. there exists an edge between any 2 pairs of nodes in V'. V' ⊆ V.

  Theorem 9.5: CLIQUE is NPC.
  IS $\leq_P$ CLIQUE.
  Given G = (V, E), construct ¬G = (V, ¬E).
  ¬G is the complement of G.
  ¬G has an edge between 2 nodes iff G does not have an edge between those 2 nodes.
  Therefore G has an independent set iff ¬G has a clique.
- **Problem Vertex Cover (VC):**
  Instance: ⟨G,k⟩ as before.
  Question: Does G have a vertex cover of size k?
  A **vertex cover** is a set of nodes (V' ⊆ V) s.t. every edge has at least one endpoint in V'.

  Theorem 9.6: VC ∈ NPC
  G = (V, E) has an IS of size K iff G = (V, E) has a VC of size n − k, where n = |V|.
  If something is an IS, its complement must be a VC.