Circuit satisfiability and Cook-Levin Theorem

Lecture 25
Thursday, April 25, 2019
25.1: Recap
**Recap**

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_p L$

$L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_p L$.

**Theorem (Cook-Levin)**

$SAT$ is **NP-Complete**.
Recap

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_P L$

$L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_P L$.

**Theorem (Cook-Levin)**

**SAT** is **NP-Complete**.
Recap

**NP**: languages that have non-deterministic polynomial time algorithms

A language \( L \) is **NP-Complete** iff

- \( L \) is in **NP**
- for every \( L' \) in **NP**, \( L' \leq_p L \)

\( L \) is **NP-Hard** if for every \( L' \) in **NP**, \( L' \leq_p L \).

**Theorem (Cook-Levin)**

**SAT** is **NP-Complete**.
Recap

**NP**: languages that have non-deterministic polynomial time algorithms

A language $L$ is **NP-Complete** iff

- $L$ is in **NP**
- for every $L'$ in **NP**, $L' \leq_p L$

$L$ is **NP-Hard** if for every $L'$ in **NP**, $L' \leq_p L$.

**Theorem (Cook-Levin)**

**SAT** is **NP-Complete**.
Pictorial View

- P
- NP
- NP-C
- NP-Hard

Chan, Har-Peled, Hassanieh (UIUC)
Possible scenarios:

1. $P = NP$.
2. $P \neq NP$

Question: Suppose $P \neq NP$. Is every problem in $NP \setminus P$ also NP-Complete?

Theorem (Ladner)

If $P \neq NP$ then there is a problem/language $X \in NP \setminus P$ such that $X$ is not NP-Complete.
Possible scenarios:

1. $P = NP$.
2. $P \neq NP$

Question: Suppose $P \neq NP$. Is every problem in $NP \setminus P$ also NP-Complete?

Theorem (Ladner)

If $P \neq NP$ then there is a problem/language $X \in NP \setminus P$ such that $X$ is not NP-Complete.
**Possible scenarios:**

1. \( P = NP \).
2. \( P \neq NP \).

**Question:** Suppose \( P \neq NP \). Is every problem in \( NP \setminus P \) also \( NP\text{-Complete} \)?

**Theorem (Ladner)**

*If \( P \neq NP \) then there is a problem/language \( X \in NP \setminus P \) such that \( X \) is not \( NP\text{-Complete} \).*
Today

NP-Completeness of three problems:

- 3-Color
- Circuit SAT

Important: understanding the problems and that they are hard.

Proofs and reductions will be sketchy and mainly to give a flavor
25.2: Circuit SAT
Circuits

Definition

A circuit is a directed *acyclic* graph with

1. Input vertices (without incoming edges) labelled with 0, 1 or a distinct variable.
2. Every other vertex is labelled \( \lor, \land \) or \( \neg \).
3. Single node output vertex with no outgoing edges.
Definition (Circuit Satisfaction (CSAT).)

Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

Claim

CSAT is in NP.

1 Certificate: Assignment to input variables.
2 Certifier: Evaluate the value of each gate in a topological sort of DAG and check the output gate value.
**CSAT**: Circuit Satisfaction

**Definition (Circuit Satisfaction (CSAT)).**

Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

**Claim**

*CSAT* is in NP.

1. **Certificate**: Assignment to input variables.
2. **Certifier**: Evaluate the value of each gate in a topological sort of DAG and check the output gate value.
Circuit SAT vs SAT

**CNF** formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas

However they are equivalent in terms of polynomial-time solvability.

**Theorem**

\[ SAT \leq_P 3SAT \leq_P CSAT. \]

**Theorem**

\[ CSAT \leq_P SAT \leq_P 3SAT. \]
CNF formulas are a rather restricted form of Boolean formulas.

Circuits are a much more powerful (and hence easier) way to express Boolean formulas.

However they are equivalent in terms of polynomial-time solvability.

**Theorem**

$$\text{SAT} \leq_p \text{3SAT} \leq_p \text{CSAT}.$$  

**Theorem**

$$\text{CSAT} \leq_p \text{SAT} \leq_p \text{3SAT}.$$
Converting a CNF formula into a Circuit

Given 3CNF formula $\varphi$ with $n$ variables and $m$ clauses, create a Circuit $C$.

- Inputs to $C$ are the $n$ boolean variables $x_1, x_2, \ldots, x_n$
- Use NOT gate to generate literal $\neg x_i$ for each variable $x_i$
- For each clause $(\ell_1 \lor \ell_2 \lor \ell_3)$ use two OR gates to mimic formula
- Combine the outputs for the clauses using AND gates to obtain the final output
Example

$3SAT \leq_p CSAT$

$\varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4)$
Converting a circuit into a **CNF** formula

Label the nodes

(A) Input circuit

(B) Label the nodes.
The other direction: \textbf{CSAT} \leq_p \text{3SAT}

1. Now: \textbf{CSAT} \leq_p \text{SAT}

Converting a circuit into a **CNF** formula

Introduce a variable for each node

(B) Label the nodes.

(C) Introduce var for each node.
Converting a circuit into a CNF formula

Write a sub-formula for each variable that is true if the var is computed correctly.

(C) Introduce var for each node.

(D) Write a sub-formula for each variable that is true if the var is computed correctly.

\[ x_k \quad \text{(Demand a sat’ assignment!)} \]
\[ x_k = x_i \land x_j \]
\[ x_j = x_g \land x_h \]
\[ x_i = \neg x_f \]
\[ x_h = x_d \lor x_e \]
\[ x_g = x_b \lor x_c \]
\[ x_f = x_a \land x_b \]
\[ x_d = 0 \]
\[ x_a = 1 \]
Converting a circuit into a **CNF** formula

Convert each sub-formula to an equivalent CNF formula.

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$x_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k = x_i \land x_j$</td>
<td>((\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \land (x_k \lor \neg x_i \lor \neg x_j))</td>
</tr>
<tr>
<td>$x_j = x_g \land x_h$</td>
<td>((\neg x_j \lor x_g) \land (\neg x_j \lor x_h) \land (x_j \lor \neg x_g \lor \neg x_h))</td>
</tr>
<tr>
<td>$x_i = \neg x_f$</td>
<td>((x_i \lor x_f) \land (\neg x_i \lor \neg x_f))</td>
</tr>
<tr>
<td>$x_h = x_d \lor x_e$</td>
<td>((x_h \lor \neg x_d) \land (x_h \lor \neg x_e) \land (\neg x_h \lor x_d \lor x_e))</td>
</tr>
<tr>
<td>$x_g = x_b \lor x_c$</td>
<td>((x_g \lor \neg x_b) \land (x_g \lor \neg x_c) \land (\neg x_g \lor x_b \lor x_c))</td>
</tr>
<tr>
<td>$x_f = x_a \land x_b$</td>
<td>((\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (x_f \lor \neg x_a \lor \neg x_b))</td>
</tr>
<tr>
<td>$x_d = 0$</td>
<td>$\neg x_d$</td>
</tr>
<tr>
<td>$x_a = 1$</td>
<td>$x_a$</td>
</tr>
</tbody>
</table>
Converting a circuit into a **CNF** formula

Take the conjunction of all the CNF sub-formulas

We got a **CNF** formula that is satisfiable if and only if the original circuit is satisfiable.
Reduction: \( \text{CSAT} \leq_p \text{SAT} \)

1. For each gate (vertex) \( v \) in the circuit, create a variable \( x_v \).

2. Case \( \neg \): \( v \) is labeled \( \neg \) and has one incoming edge from \( u \) (so \( x_v = \neg x_u \)). In \text{SAT} formula generate, add clauses \((x_u \lor x_v)\), \((\neg x_u \lor \neg x_v)\). Observe that

\[
x_v = \neg x_u \text{ is true } \iff \begin{cases} x_u \lor x_v \text{ both true.} \\
(\neg x_u \lor \neg x_v) \text{ both true.}
\end{cases}
\]
Case $\lor$: So $x_v = x_u \lor x_w$. In SAT formula generated, add clauses $(x_v \lor \neg x_u)$, $(x_v \lor \neg x_w)$, and $(\neg x_v \lor x_u \lor x_w)$. Again, observe that

$$(x_v = x_u \lor x_w) \text{ is true } \iff (x_v \lor \neg x_u), \quad (x_v \lor \neg x_w), \quad (\neg x_v \lor x_u \lor x_w) \text{ all true.}$$
Reduction: \( \text{CSAT} \leq_{P} \text{SAT} \)

Continued...

\( \text{Case } \land: \) So \( x_v = x_u \land x_w \). In \( \text{SAT} \) formula generated, add clauses \((\neg x_v \lor x_u)\), \((\neg x_v \lor x_w)\), and \((x_v \lor \neg x_u \lor \neg x_w)\).

Again observe that

\[
x_v = x_u \land x_w \text{ is true } \iff (\neg x_v \lor x_u), (\neg x_v \lor x_w), (x_v \lor \neg x_u \lor \neg x_w) \text{ all true.}
\]
Reduction: \( \text{CSAT} \leq_p \text{SAT} \)

Continued...

1. If \( v \) is an input gate with a fixed value then we do the following. If \( x_v = 1 \) add clause \( x_v \). If \( x_v = 0 \) add clause \( \neg x_v \).

2. Add the clause \( x_v \) where \( v \) is the variable for the output gate.
Correctness of Reduction

Need to show circuit $C$ is satisfiable iff $\varphi_C$ is satisfiable

$\Rightarrow$ Consider a satisfying assignment $a$ for $C$

1. Find values of all gates in $C$ under $a$
2. Give value of gate $v$ to variable $x_v$; call this assignment $a'$
3. $a'$ satisfies $\varphi_C$ (exercise)

$\Leftarrow$ Consider a satisfying assignment $a$ for $\varphi_C$

1. Let $a'$ be the restriction of $a$ to only the input variables
2. Value of gate $v$ under $a'$ is the same as value of $x_v$ in $a$
3. Thus, $a'$ satisfies $C$
List of NP-Complete Problems to Remember

Problems

1. SAT
2. 3SAT
3. CircuitSAT
4. Independent Set
5. Clique
6. Vertex Cover
7. Hamilton Cycle and Hamilton Path in both directed and undirected graphs
8. 3Color and Color
25.3: NP-Completeness of Graph Coloring
**Problem:** Graph Coloring

**Instance:** \( G = (V, E) \): Undirected graph, integer \( k \).

**Question:** Can the vertices of the graph be colored using \( k \) colors so that vertices connected by an edge do not get the same color?
Graph 3-Coloring

Problem: 3 Coloring

Instance: $G = (V, E)$: Undirected graph.
Question: Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?
Graph 3-Coloring

Problem: 3 Coloring

**Instance:** $G = (V, E)$: Undirected graph.

**Question:** Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?
1. Observation: If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

2. $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

3. Graph 2-Coloring can be decided in polynomial time.

4. $G$ is 2-colorable iff $G$ is bipartite!

5. There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).
Observation: If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

$G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

Graph 2-Coloring can be decided in polynomial time.

$G$ is 2-colorable iff $G$ is bipartite!

There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).
1. **Observation**: If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

2. $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

3. Graph 2-Coloring can be decided in polynomial time.

4. $G$ is 2-colorable iff $G$ is bipartite!

5. There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).
Graph Coloring

1. **Observation**: If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

2. $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

3. Graph 2-Coloring can be decided in polynomial time.

4. $G$ is 2-colorable iff $G$ is bipartite!

5. There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).
Graph Coloring

1. **Observation:** If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

2. $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

3. Graph $2$-Coloring can be decided in polynomial time.

4. $G$ is $2$-colorable iff $G$ is bipartite!

5. There is a linear time algorithm to check if $G$ is bipartite using **BFS** (we saw this earlier).
Graph Coloring

1. **Observation**: If $G$ is colored with $k$ colors then each color class (nodes of same color) form an independent set in $G$.

2. $G$ can be partitioned into $k$ independent sets iff $G$ is $k$-colorable.

3. Graph 2-Coloring can be decided in polynomial time.

4. $G$ is 2-colorable iff $G$ is bipartite!

5. There is a linear time algorithm to check if $G$ is bipartite using BFS (we saw this earlier).
25.3.1: Problems related to graph coloring
Register Allocation

Assign variables to (at most) $k$ registers such that variables needed at the same time are not assigned to the same register.

Interference Graph

Vertices are variables, and there is an edge between two vertices, if the two variables are “live” at the same time.

Observations

- [Chaitin] Register allocation problem is equivalent to coloring the interference graph with $k$ colors.
- Moreover, $3$-COLOR $\leq_P k$-Register Allocation, for any $k \geq 3$. 
Class Room Scheduling

1. Given $n$ classes and their meeting times, are $k$ rooms sufficient?
2. Reduce to Graph $k$-Coloring problem
3. Create graph $G$
   - a node $v_i$ for each class $i$
   - an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict
4. Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.
Class Room Scheduling

1. Given $n$ classes and their meeting times, are $k$ rooms sufficient?
2. Reduce to Graph $k$-Coloring problem
3. Create graph $G$
   - a node $v_i$ for each class $i$
   - an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict
4. Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.
Class Room Scheduling

1. Given $n$ classes and their meeting times, are $k$ rooms sufficient?
2. Reduce to Graph $k$-Coloring problem
3. Create graph $G$
   - a node $v_i$ for each class $i$
   - an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict
4. Exercise: $G$ is $k$-colorable iff $k$ rooms are sufficient.
Class Room Scheduling

1. Given \( n \) classes and their meeting times, are \( k \) rooms sufficient?
2. Reduce to Graph \( k \)-Coloring problem
3. Create graph \( G \)
   - a node \( v_i \) for each class \( i \)
   - an edge between \( v_i \) and \( v_j \) if classes \( i \) and \( j \) conflict
4. Exercise: \( G \) is \( k \)-colorable iff \( k \) rooms are sufficient.
Class Room Scheduling

1. Given *n* classes and their meeting times, are *k* rooms sufficient?
2. Reduce to Graph *k*-Coloring problem
3. Create graph $G$
   - a node $v_i$ for each class $i$
   - an edge between $v_i$ and $v_j$ if classes $i$ and $j$ conflict
4. Exercise: $G$ is *k*-colorable iff *k* rooms are sufficient.
Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA)

- Breakup a frequency range \([a, b]\) into disjoint bands of frequencies \([a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]\)
- Each cell phone tower (simplifying) gets one band
- Constraint: nearby towers cannot be assigned the same band, otherwise signals will interfere

Problem: given \(k\) bands and some region with \(n\) towers, is there a way to assign the bands to avoid interference?

Can reduce to \(k\)-coloring by creating interference/conflict graph on towers.
Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA)

- Breakup a frequency range $[a, b]$ into disjoint bands of frequencies $[a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]$
- Each cell phone tower (simplifying) gets one band
- Constraint: nearby towers cannot be assigned same band, otherwise signals will interfere

**Problem:** given $k$ bands and some region with $n$ towers, is there a way to assign the bands to avoid interference?

Can reduce to $k$-coloring by creating interference/conflict graph on towers.
Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA)

- Breakup a frequency range \([a, b]\) into disjoint \textit{bands} of frequencies \([a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]\)
- Each cell phone tower (simplifying) gets one band
- Constraint: nearby towers cannot be assigned same band, otherwise signals will interference

**Problem:** given \(k\) bands and some region with \(n\) towers, is there a way to assign the bands to avoid interference?

**Can reduce to** \(k\)-coloring by creating interference/conflict graph on towers.
25.4: Showing hardness of 3 COLORING
3 color this gadget.

Clicker question

You are given three colors: red, green and blue. Can the following graph be three colored in a valid way (assuming the two nodes are already colored as indicated).

\[ \text{Yes.} \]
\[ \text{No.} \]
You are given three colors: red, green and blue. Can the following graph be three colored in a valid way (assuming the two nodes are already colored as indicated).

Yes.  
No.
3-Coloring is **NP-Complete**

- **3-Coloring** is in **NP**.
  - **Certificate**: for each node a color from \( \{1, 2, 3\} \).
  - **Certifier**: Check if for each edge \((u, v)\), the color of \(u\) is different from that of \(v\).

- **Hardness**: We will show **3-SAT \( \leq_p \) 3-Coloring**.
Reduction Idea

1. **ϕ**: Given \textbf{3SAT} formula (i.e., \textbf{3CNF} formula).
2. **ϕ**: variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$.
3. Create graph $G_\varphi$ s.t. $G_\varphi$ 3-colorable $\iff \varphi$ satisfiable.
   - encode assignment $x_1, \ldots, x_n$ in colors assigned nodes of $G_\varphi$.
   - create triangle with node True, False, Base.
   - for each variable $x_i$ two nodes $v_i$ and $\bar{v}_i$ connected in a triangle with common Base.
   - If graph is 3-colored, either $v_i$ or $\bar{v}_i$ gets the same color as True. Interpret this as a truth assignment to $v_i$.
   - Need to add constraints to ensure clauses are satisfied (next phase).
Reduction Idea

1. $\phi$: Given 3SAT formula (i.e., 3CNF formula).
2. $\phi$: variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$.
3. Create graph $G_\phi$ s.t. $G_\phi$ 3-colorable $\iff \phi$ satisfiable.
   - encode assignment $x_1, \ldots, x_n$ in colors assigned nodes of $G_\phi$.
   - create triangle with node True, False, Base
   - for each variable $x_i$ two nodes $v_i$ and $\bar{v}_i$ connected in a triangle with common Base
   - If graph is 3-colored, either $v_i$ or $\bar{v}_i$ gets the same color as True. Interpret this as a truth assignment to $v_i$
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. \( \varphi \): Given \textbf{3SAT} formula (i.e., \textbf{3CNF} formula).
2. \( \varphi \): variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \).
3. Create graph \( G_\varphi \) s.t. \( G_\varphi \) 3-colorable \iff \( \varphi \) satisfiable.
   - encode assignment \( x_1, \ldots, x_n \) in colors assigned nodes of \( G_\varphi \).
   - create triangle with node True, False, Base
   - for each variable \( x_i \) two nodes \( v_i \) and \( \bar{v}_i \) connected in a triangle with common Base
   - If graph is 3-colored, either \( v_i \) or \( \bar{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \)
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. $\varphi$: Given 3SAT formula (i.e., 3CNF formula).
2. $\varphi$: variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$.
3. Create graph $G_\varphi$ s.t. $G_\varphi$ 3-colorable $\iff$ $\varphi$ satisfiable.
   - encode assignment $x_1, \ldots, x_n$ in colors assigned nodes of $G_\varphi$.
   - create triangle with node True, False, Base
   - for each variable $x_i$ two nodes $v_i$ and $\bar{v}_i$ connected in a triangle with common Base
   - If graph is 3-colored, either $v_i$ or $\bar{v}_i$ gets the same color as True. Interpret this as a truth assignment to $v_i$
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. \( \varphi \): Given \textbf{3SAT} formula (i.e., \textbf{3CNF} formula).
2. \( \varphi \): variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \).
3. Create graph \( G_\varphi \) s.t. \( G_\varphi \) 3-colorable \iff \( \varphi \) satisfiable.
   - encode assignment \( x_1, \ldots, x_n \) in colors assigned nodes of \( G_\varphi \).
   - create triangle with node True, False, Base
   - for each variable \( x_i \) two nodes \( v_i \) and \( \bar{v}_i \) connected in a triangle with common Base
   - If graph is 3-colored, either \( v_i \) or \( \bar{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \)
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. \( \varphi \): Given **3SAT** formula (i.e., **3CNF** formula).
2. \( \varphi \): variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \).
3. Create graph \( G_\varphi \) s.t. \( G_\varphi \ 3\text{-colorable} \iff \varphi \text{ satisfiable.} \)
   - encode assignment \( x_1, \ldots, x_n \) in colors assigned nodes of \( G_\varphi \).
   - create triangle with node True, False, Base
   - for each variable \( x_i \) two nodes \( v_i \) and \( \bar{v}_i \) connected in a triangle with common Base
   - If graph is 3-colored, either \( v_i \) or \( \bar{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \)
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. \( \varphi \): Given **3SAT** formula (i.e., **3CNF** formula).
2. \( \varphi \): variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \).
3. Create graph \( G_\varphi \) s.t. \( G_\varphi \) 3-colorable \( \iff \varphi \) satisfiable.
   - encode assignment \( x_1, \ldots, x_n \) in colors assigned nodes of \( G_\varphi \).
   - create triangle with node True, False, Base
   - for each variable \( x_i \) two nodes \( v_i \) and \( \bar{v}_i \) connected in a triangle with common Base
   - If graph is 3-colored, either \( v_i \) or \( \bar{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \)
   - Need to add constraints to ensure clauses are satisfied (next phase)
Reduction Idea

1. \( \varphi \): Given \textbf{3SAT} formula (i.e., \textbf{3CNF} formula).
2. \( \varphi \): variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \).
3. Create graph \( G_\varphi \) s.t. \( G_\varphi \) 3-colorable \( \iff \varphi \) satisfiable.
   - encode assignment \( x_1, \ldots, x_n \) in colors assigned nodes of \( G_\varphi \).
   - create triangle with node True, False, Base
   - for each variable \( x_i \) two nodes \( v_i \) and \( \overline{v}_i \) connected in a triangle with common Base
   - If graph is 3-colored, either \( v_i \) or \( \overline{v}_i \) gets the same color as True. Interpret this as a truth assignment to \( v_i \)
   - Need to add constraints to ensure clauses are satisfied (next phase)
Clause Satisfiability Gadget

1. For each clause $C_j = (a \lor b \lor c)$, create a small gadget graph
   - gadget graph connects to nodes corresponding to $a, b, c$
   - needs to implement OR

2. OR-gadget-graph:
1. For each clause $C_j = (a \lor b \lor c)$, create a small gadget graph:
   - gadget graph connects to nodes corresponding to $a, b, c$
   - needs to implement OR

2. OR-gadget-graph:
Property: if $a, b, c$ are colored False in a 3-coloring then output node of OR-gadget has to be colored False.

Property: if one of $a, b, c$ is colored True then OR-gadget can be 3-colored such that output node of OR-gadget is colored True.
Reduction

- create triangle with nodes True, False, Base
- for each variable $x_i$ two nodes $v_i$ and $\overline{v}_i$ connected in a triangle with common Base
- for each clause $C_j = (a \lor b \lor c)$, add OR-gadget graph with input nodes $a, b, c$ and connect output node of gadget to both False and Base

![Diagram](https://example.com/diagram.png)
Claim

No legal $3$-coloring of above graph (with coloring of nodes $T$, $F$, $B$ fixed) in which $a, b, c$ are colored False. If any of $a, b, c$ are colored True then there is a legal $3$-coloring of above graph.
3 coloring of the clause gadget

FFF - BAD

FTT

TFF

TTF

TTF

TFT

FFF - BAD
Example

\[ \varphi = (u \lor \neg v \lor w) \land (v \lor x \lor \neg y) \]

Variable and negations have complementary colors. Literals get colors T or F.

Palette

or gates
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- if \( x_i \) is assigned True, color \( v_i \) True and \( \overline{v}_i \) False
- for each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- if \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment
- consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Correctness of Reduction

φ is satisfiable implies $G_\varphi$ is 3-colorable
- if $x_i$ is assigned True, color $v_i$ True and $\bar{v}_i$ False
- for each clause $C_j = (a \lor b \lor c)$ at least one of $a, b, c$ is colored True. OR-gadget for $C_j$ can be 3-colored such that output is True.

$G_\varphi$ is 3-colorable implies φ is satisfiable
- if $v_i$ is colored True then set $x_i$ to be True, this is a legal truth assignment
- consider any clause $C_j = (a \lor b \lor c)$, it cannot be that all $a, b, c$ are False. If so, output of OR-gadget for $C_j$ has to be colored False but output is connected to Base and False!
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- if \( x_i \) is assigned True, color \( v_i \) True and \( \bar{v}_i \) False
- for each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- if \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment
- consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- if \( x_i \) is assigned True, color \( v_i \) True and \( \bar{v}_i \) False
- for each clause \( C_j = (a \lor b \lor c) \) at least one of \( a \), \( b \), \( c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- if \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment
- consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a \), \( b \), \( c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Correctness of Reduction

\( \varphi \) is satisfiable implies \( G_\varphi \) is 3-colorable

- if \( x_i \) is assigned True, color \( v_i \) True and \( \bar{v}_i \) False
- for each clause \( C_j = (a \lor b \lor c) \) at least one of \( a, b, c \) is colored True. OR-gadget for \( C_j \) can be 3-colored such that output is True.

\( G_\varphi \) is 3-colorable implies \( \varphi \) is satisfiable

- if \( v_i \) is colored True then set \( x_i \) to be True, this is a legal truth assignment
- consider any clause \( C_j = (a \lor b \lor c) \). it cannot be that all \( a, b, c \) are False. If so, output of OR-gadget for \( C_j \) has to be colored False but output is connected to Base and False!
Graph generated in reduction...
... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
Graph generated in reduction...

... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
Graph generated in reduction...

... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
Graph generated in reduction...
... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor c \lor \overline{d}) \land (a \lor c \lor d) \land (a \lor b \lor \overline{d})\]
Graph generated in reduction...

... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
Graph generated in reduction...
... from 3SAT to 3COLOR

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
25.5: Proof of Cook-Levin Theorem
Cook-Levin Theorem

Theorem (Cook-Levin)

\textit{SAT} is \textit{NP}-Complete.

We have already seen that \textit{SAT} is in \textit{NP}.

Need to prove that every language $L \in \text{NP}$, $L \leq_P \text{SAT}$

\textbf{Difficulty:} Infinite number of languages in \textit{NP}. Must \textit{simultaneously} show a \textit{generic} reduction strategy.
Theorem (Cook-Levin)

**SAT** is NP-Complete.

We have already seen that **SAT** is in **NP**.

Need to prove that every language $L \in \text{NP}$, $L \leq_P \text{SAT}$

**Difficulty:** Infinite number of languages in **NP**. Must *simultaneously* show a *generic* reduction strategy.
High-level Plan

What does it mean that $L \in \text{NP}$?

$L \in \text{NP}$ implies that there is a non-deterministic TM $M$ and polynomial $p()$ such that

$$L = \{ x \in \Sigma^* \mid M \text{ accepts } x \text{ in at most } p(|x|) \text{ steps} \}$$

We will describe a reduction $f_M$ that depends on $M, p$ such that:

- $f_M$ takes as input a string $x$ and outputs a SAT formula $f_M(x)$
- $f_M$ runs in time polynomial in $|x|$
- $x \in L$ if and only if $f_M(x)$ is satisfiable
What does it mean that $L \in \text{NP}$?

$L \in \text{NP}$ implies that there is a non-deterministic TM $M$ and polynomial $p()$ such that

$$L = \{x \in \Sigma^* \mid M \text{ accepts } x \text{ in at most } p(|x|) \text{ steps}\}$$

We will describe a reduction $f_M$ that depends on $M, p$ such that:

- $f_M$ takes as input a string $x$ and outputs a SAT formula $f_M(x)$
- $f_M$ runs in time polynomial in $|x|$
- $x \in L$ if and only if $f_M(x)$ is satisfiable
$f_M(x)$ is satisfiable if and only if $x \in L$

$f_M(x)$ is satisfiable if and only if nondeterministic $M$ accepts $x$ in $p(|x|)$ steps

**BIG IDEA**

- $f_M(x)$ will express “$M$ on input $x$ accepts in $p(|x|)$ steps”
- $f_M(x)$ will encode a computation history of $M$ on $x$
- $f_M(x)$ will be a carefully constructed CNF formula s.t if we have a satisfying assignment to it, then we will be able to see a complete accepting computation of $M$ on $x$ *down to the last detail* of where the head is, what transition is chosen, what the tape contents are, at each step.
Plan continued

\[ f_M(x) \] is satisfiable if and only if \( x \in L \)

\[ f_M(x) \] is satisfiable if and only if nondeterministic \( M \) accepts \( x \) in \( p(|x|) \) steps

**BIG IDEA**

- \( f_M(x) \) will express “\( M \) on input \( x \) accepts in \( p(|x|) \) steps”
- \( f_M(x) \) will encode a computation history of \( M \) on \( x \)
- \( f_M(x) \) will be a carefully constructed **CNF** formula s.t if we have a satisfying assignment to it, then we will be able to see a complete accepting computation of \( M \) on \( x \) *down to the last detail* of where the head is, what transition is chosen, what the tape contents are, at each step.
**Tableau of Computation**

\( M \) runs in time \( p(|x|) \) on \( x \). Entire computation of \( M \) on \( x \) can be represented by a “tableau”

Row \( i \) gives contents of all cells at time \( i \)
At time 0 tape has input \( x \) followed by blanks
Each row long enough to hold all cells \( M \) might ever have scanned.
Variable of \( f_M(x) \)

Four types of variable to describe computation of \( M \) on \( x \):

- \( T(b, h, i) \): tape cell at position \( h \) holds symbol \( b \) at time \( i \).
  \[ 1 \leq h \leq p(|x|), \ b \in \Gamma, \ 0 \leq i \leq p(|x|) \]

- \( H(h, i) \): read/write head is at position \( h \) at time \( i \).
  \[ 1 \leq h \leq p(|x|), \ 0 \leq i \leq p(|x|) \]

- \( S(q, i) \): state of \( M \) is \( q \) at time \( i \).
  \( q \in Q, \ 0 \leq i \leq p(|x|) \)

- \( I(j, i) \): instruction number \( j \) is executed at time \( i \)

\( M \) is non-deterministic, need to specify transitions in some way.
Number transitions as \( 1, 2, \ldots, \ell \) where \( j \)th transition is
\( < q_j, b_j, q'_j, b'_j, d_j > \) indication \((q'_j, b'_j, d_j) \in \delta(q_j, b_j), \)
direction \( d_j \in \{-1, 0, 1\} \).

Number of variables is \( O(p(|x|)^2) \) where constant in \( O() \) hides
dependence on fixed machine \( M \).
Notation

Some abbreviations for ease of notation
\[ \bigwedge_{k=1}^{m} x_k \text{ means } x_1 \land x_2 \land \ldots \land x_m \]

\[ \bigvee_{k=1}^{m} x_k \text{ means } x_1 \lor x_2 \lor \ldots \lor x_m \]

\[ \bigoplus (x_1, x_2, \ldots, x_k) \] is a formula that means exactly one of \( x_1, x_2, \ldots, x_m \) is true. Can be converted to \text{CNF} \text{ form}
Clauses of $f_M(x)$

$f_M(x)$ is the conjunction of 8 clause groups:

$$f_M(x) = \varphi_1 \land \varphi_2 \land \varphi_3 \land \varphi_4 \land \varphi_5 \land \varphi_6 \land \varphi_7 \land \varphi_8$$

where each $\varphi_i$ is a CNF formula. Described in subsequent slides.

**Property:** $f_M(x)$ is satisfied iff there is a truth assignment to the variables that simultaneously satisfy $\varphi_1, \ldots, \varphi_8$. 
\( \varphi_1 \) asserts (is true iff) the variables are set T/F indicating that \( M \) starts in state \( q_0 \) at time \( 0 \) with tape contents containing \( x \) followed by blanks.

Let \( x = a_1 a_2 \ldots a_n \)

\[
\varphi_1 = S(q, 0) \quad \text{state at time 0 is } q_0
\]

\[
\land \quad \text{and}
\]

\[
\land_{h=1}^{n} T(a_h, h, 0) \quad \text{at time 0 cells } 1 \text{ to } n \text{ have } a_1 \text{ to } a_n
\]

\[
\land_{h=n+1}^{p(|x|)} T(B, h, 0) \quad \text{at time 0 cells } n + 1 \text{ to } p(|x|) \text{ have blanks}
\]

\[
\land \quad \text{and}
\]

\[
H(1, 0) \quad \text{head at time 0 is in position } 1
\]
$\varphi_2$ asserts $M$ in exactly one state at any time $i$

$\varphi_2 = \bigwedge_{i=0}^{p(|x|)} \left( \bigoplus (S(q_0, i), S(q_1, i), \ldots, S(q_{|Q|}, i)) \right)$
$\varphi_3$ asserts that each tape cell holds a unique symbol at any given time.

$$\varphi_3 = \bigwedge_{i=0}^{p(|x|)} \bigwedge_{h=1}^{p(|x|)} \bigoplus(T(b_1, h, i), T(b_2, h, i), \ldots, T(b_{|\Gamma|}, h, i))$$

For each time $i$ and for each cell position $h$ exactly one symbol $b \in \Gamma$ at cell position $h$ at time $i$
\( \varphi_4 \) asserts that the read/write head of \( M \) is in exactly one position at any time \( i \)

\[
\varphi_4 = \bigwedge_{i=0}^{p(|x|)} \left( \bigoplus (H(1, i), H(2, i), \ldots, H(p(|x|), i)) \right)
\]
$\varphi_5$ asserts that $M$ accepts

- Let $q_a$ be unique accept state of $M$
- without loss of generality assume $M$ runs all $p(|x|)$ steps

$$\varphi_5 = S(q_a, p(|x|))$$

State at time $p(|x|)$ is $q_a$ the accept state.

If we don’t want to make assumption of running for all steps

$$\varphi_5 = \bigvee_{i=1}^{p(|x|)} S(q_a, i)$$

which means $M$ enters accepts state at some time.
$\varphi_6$ asserts that $M$ executes a unique instruction at each time

$$
\varphi_6 = \bigwedge_{i=0}^{p(|x|)} \bigoplus (I(1, i), I(2, i), \ldots, I(m, i))
$$

where $m$ is max instruction number.
\( \varphi_7 \) ensures that variables don’t allow tape to change from one moment to next if the read/write head was not there.

“If head is not at position \( h \) at time \( i \) then at time \( i + 1 \) the symbol at cell \( h \) must be unchanged”

\[
\varphi_7 = \bigwedge_i \bigwedge_h \bigwedge_{b \neq c} \left( H(h, i) \Rightarrow T(b, h, i) \land T(c, h, i + 1) \right)
\]

since \( A \Rightarrow B \) is same as \( \neg A \lor B \), rewrite above in CNF form

\[
\varphi_7 = \bigwedge_i \bigwedge_h \bigwedge_{b \neq c} \left( H(h, i) \lor \neg T(b, h, i) \lor \neg T(c, h, i + 1) \right)
\]
\( \varphi_8 \) asserts that changes in tableau/tape correspond to transitions of \( M \) (as Lenny says, this is the big cookie).

Let \( j \)th instruction be \( \langle q_j, b_j, q'_j, b'_j, d_j \rangle \)

\[
\varphi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i)) \quad \text{If instr } j \text{ executed at time } i \text{ then state must be correct to do } j \\
\bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1)) \quad \text{and at next time unit, state must be the proper next state for instr } j \\
\bigwedge_i \bigwedge_h \bigwedge_j [(I(j, i) \land H(h, i)) \Rightarrow T(b_j, h, i)] \quad \text{if } j \text{ was executed and head was at position } h, \text{ then cell } h \text{ has correct symbol for } j \\
\bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \land H(h, i)) \Rightarrow T(b'_j, h, i + 1)] \quad \text{if } j \text{ was done then at time } i \text{ with head at } h \text{ then at next time step symbol } b'_j \text{ was indeed written in position } h \\
\bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \land H(h, i)) \Rightarrow H(h + d_j, i + 1)] \quad \text{and head is moved properly according to instr } j.
Proof of Correctness

(Sketch)

- Given $M$, $x$, poly-time algorithm to construct $f_M(x)$
- if $f_M(x)$ is satisfiable then the truth assignment completely specifies an accepting computation of $M$ on $x$
- if $M$ accepts $x$ then the accepting computation leads to an ”obvious” truth assignment to $f_M(x)$. Simply assign the variables according to the state of $M$ and cells at each time $i$.

Thus $M$ accepts $x$ if and only if $f_M(x)$ is satisfiable