Circuit satisfiability and Cook-Levin Theorem

Lecture 24
Thursday, December 3, 2020
24.1 Recap
Recap

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

A language $L$ is **NP-Complete** if and only if

- $L$ is in $\text{NP}$
- for every $L'$ in $\text{NP}$, $L' \leq_P L$

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

**Theorem 24.1 (Cook-Levin).**

$\text{SAT}$ is NP-Complete.
Recap

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

A language $L$ is **NP-Complete** if and only if

- $L$ 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 24.1 (Cook-Levin).**

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

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

A language $L$ is **NP-Complete** if and only if

- $L$ 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 24.1 (Cook-Levin).**

**SAT** is NP-Complete.
Recap

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

A language \( L \) is **NP-Complete** if and only if

- \( L \) 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 24.1 (Cook-Levin).**

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

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 24.2 (Ladner).

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

Possible scenarios:

1. \( P = \text{NP} \).
2. \( P \neq \text{NP} \)

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

**Theorem 24.2 (Ladner).**

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

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 24.2 (Ladner).

If $P \neq NP$ then there is a problem/language $X \in NP \setminus P$ such that $X$ is not NP-Complete.
What do we know so far

1. **Independent Set** $\leq_p$ **Clique**, **Clique** $\leq_p$ **Independent Set**.
   $\implies$ **Clique** $\approx_p$ **Independent Set**.

2. **Vertex Cover** $\leq_p$ **Independent Set**, **Independent Set** $\leq_p$ **Vertex Cover**.
   $\implies$ **Independent Set** $\approx_p$ **Vertex Cover**.

3. **3SAT** $\leq_p$ **SAT**, **SAT** $\leq_p$ **3SAT** $\implies$ **3SAT** $\approx_p$ **SAT**.

4. **3SAT** $\leq_p$ **Independent Set**.
   Exercise (or Cook-Levin theorem): **Independent Set** $\leq_p$ **SAT**
   $\implies$ **3SAT** $\approx_p$ **Independent Set**.

5. **SAT** $\leq_p$ **Hamiltonian Cycle**
   Exercise (or Cook-Levin theorem): **Hamiltonian Cycle** $\leq_p$ **3SAT**
   $\implies$ **Hamiltonian Cycle** $\approx_p$ **3SAT**

6. **Clique** $\approx_p$ **Independent Set** $\approx_p$ **Vertex Cover** $\approx_p$ **3SAT**
   $\approx_p$ **SAT** $\approx_p$ **Hamiltonian Cycle**
What do we know so far

1. **Independent Set** \(\leq_P\) **Clique**, **Clique** \(\leq_P\) **Independent Set**.
   \[\implies\] **Clique** \(\approx_P\) **Independent Set**.

2. **Vertex Cover** \(\leq_P\) **Independent Set**, **Independent Set** \(\leq_P\) **Vertex Cover**.
   \[\implies\] **Independent Set** \(\approx_P\) **Vertex Cover**.

3. **3SAT** \(\leq_P\) **SAT**, **SAT** \(\leq_P\) **3SAT** \[\implies\] **3SAT** \(\approx_P\) **SAT**.

4. **3SAT** \(\leq_P\) **Independent Set**.
   Exercise (or Cook-Levin theorem): **Independent Set** \(\leq_P\) **SAT**
   \[\implies\] **3SAT** \(\approx_P\) **Independent Set**.

5. **SAT** \(\leq_P\) **Hamiltonian Cycle**
   Exercise (or Cook-Levin theorem): **Hamiltonian Cycle** \(\leq_P\) **3SAT**
   \[\implies\] **Hamiltonian Cycle** \(\approx_P\) **3SAT**

6. **Clique** \(\approx_P\) **Independent Set** \(\approx_P\) **Vertex Cover** \(\approx_P\) **3SAT**
   \(\approx_P\) **SAT** \(\approx_P\) **Hamiltonian Cycle**
What do we know so far

1. Independent Set $\leq_P$ Clique, Clique $\leq_P$ Independent Set.
   $\implies$ Clique $\approx_P$ Independent Set.

2. Vertex Cover $\leq_P$ Independent Set, Independent Set $\leq_P$ Vertex Cover.
   $\implies$ Independent Set $\approx_P$ Vertex Cover.

3. 3SAT $\leq_P$ SAT, SAT $\leq_P$ 3SAT $\implies$ 3SAT $\approx_P$ SAT.

4. 3SAT $\leq_P$ Independent Set.
   Exercise (or Cook-Levin theorem): Independent Set $\leq_P$ SAT
   $\implies$ 3SAT $\approx_P$ Independent Set.

5. SAT $\leq_P$ Hamiltonian Cycle
   Exercise (or Cook-Levin theorem): Hamiltonian Cycle $\leq_P$ 3SAT
   $\implies$ Hamiltonian Cycle $\approx_P$ 3SAT

6. Clique $\approx_P$ Independent Set $\approx_P$ Vertex Cover $\approx_P$ 3SAT
   $\approx_P$ SAT $\approx_P$ Hamiltonian Cycle
What do we know so far

1. **Independent Set** $\leq_P$ **Clique**, **Clique** $\leq_P$ **Independent Set**.
   $\Rightarrow$ **Clique** $\cong_P$ **Independent Set**.

2. **Vertex Cover** $\leq_P$ **Independent Set**, **Independent Set** $\leq_P$ **Vertex Cover**.
   $\Rightarrow$ **Independent Set** $\cong_P$ **Vertex Cover**.

3. **3SAT** $\leq_P$ **SAT**, **SAT** $\leq_P$ **3SAT** $\Rightarrow$ **3SAT** $\cong_P$ **SAT**.

4. **3SAT** $\leq_P$ **Independent Set**.
   Exercise (or Cook-Levin theorem): **Independent Set** $\leq_P$ **SAT**
   $\Rightarrow$ **3SAT** $\cong_P$ **Independent Set**.

5. **SAT** $\leq_P$ **Hamiltonian Cycle**
   Exercise (or Cook-Levin theorem): **Hamiltonian Cycle** $\leq_P$ **3SAT**
   $\Rightarrow$ **Hamiltonian Cycle** $\cong_P$ **3SAT**

6. **Clique** $\cong_P$ **Independent Set** $\cong_P$ **Vertex Cover** $\cong_P$ **3SAT** $\cong_P$ **SAT** $\cong_P$ **Hamiltonian Cycle**
What do we know so far

1. Independent Set $\leq_p$ Clique, Clique $\leq_p$ Independent Set.\
   $\implies$ Clique $\approx_p$ Independent Set.

2. Vertex Cover $\leq_p$ Independent Set, Independent Set $\leq_p$ Vertex Cover.\
   $\implies$ Independent Set $\approx_p$ Vertex Cover.

3. 3SAT $\leq_p$ SAT, SAT $\leq_p$ 3SAT $\implies$ 3SAT $\approx_p$ SAT.

4. 3SAT $\leq_p$ Independent Set.
   Exercise (or Cook-Levin theorem): Independent Set $\leq_p$ SAT\
   $\implies$ 3SAT $\approx_p$ Independent Set.

5. SAT $\leq_p$ Hamiltonian Cycle
   Exercise (or Cook-Levin theorem): Hamiltonian Cycle $\leq_p$ 3SAT\
   $\implies$ Hamiltonian Cycle $\approx_p$ 3SAT

6. Clique $\approx_p$ Independent Set $\approx_p$ Vertex Cover $\approx_p$ 3SAT\
   $\approx_p$ SAT $\approx_p$ Hamiltonian Cycle
What do we know so far

1. Independent Set $\leq_P$ Clique, Clique $\leq_P$ Independent Set.
   $\implies$ Clique $\cong_P$ Independent Set.

2. Vertex Cover $\leq_P$ Independent Set, Independent Set $\leq_P$ Vertex Cover.
   $\implies$ Independent Set $\cong_P$ Vertex Cover.

3. 3SAT $\leq_P$ SAT, SAT $\leq_P$ 3SAT $\implies$ 3SAT $\cong_P$ SAT.

4. 3SAT $\leq_P$ Independent Set.
   Exercise (or Cook-Levin theorem): Independent Set $\leq_P$ SAT
   $\implies$ 3SAT $\cong_P$ Independent Set.

5. SAT $\leq_P$ Hamiltonian Cycle
   Exercise (or Cook-Levin theorem): Hamiltonian Cycle $\leq_P$ 3SAT
   $\implies$ Hamiltonian Cycle $\cong_P$ 3SAT

6. Clique $\cong_P$ Independent Set $\cong_P$ Vertex Cover $\cong_P$ 3SAT
   $\cong_P$ SAT $\cong_P$ Hamiltonian Cycle
NP Completeness

Clique $\approx_P$ Independent Set $\approx_P$ Vertex Cover $\approx_P$ 3SAT $\approx_P$ SAT $\approx_P$ SAT $\approx_P$ Hamiltonian Cycle

All these problems are in NP.

SAT is NPC.

All these problems are NP-Complete.
NP Completeness

Clique $\approx_P$ Independent Set $\approx_P$ Vertex Cover $\approx_P$ 3SAT $\approx_P$ SAT $\approx_P$ Hamiltonian Cycle

All these problems are in NP.

SAT is NPC.

All these problems are NP-Complete.
NP Completeness

Clique $\equiv_P$ Independent Set $\equiv_P$ Vertex Cover $\equiv_P$ 3SAT $\equiv_P$ SAT $\equiv_P$ Hamiltonian Cycle

All these problems are in $\text{NP}$.

$\text{SAT}$ is $\text{NPC}$.

All these problems are $\text{NP-Complete}$.
NP Completeness

Clique $\cong_P$ Independent Set $\cong_P$ Vertex Cover $\cong_P$ 3SAT $\cong_P$ SAT $\cong_P$ Hamiltonian Cycle

All these problems are in $\textbf{NP}$.

$\textbf{SAT}$ is $\textbf{NPC}$.

All these problems are $\textbf{NP-Complete}$.
THE END

... (for now)
24.2
Circuit SAT
24.2.1
The circuit satisfiability (CSAT) problem
Definition 24.1.
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.

Can safely assume every node has at most two incoming edges.
Circuits

**Definition 24.1.**

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.

Can safely assume every node has at most two incoming edges.
**CSAT**: Circuit Satisfaction

**Definition 24.2 (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 24.3.**

*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 24.2 (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 24.3.**

*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.
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.
Converting a CNF formula into a Circuit

\[ 3\text{SAT} \leq_P \text{CSAT} \]

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

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

$$\varphi = \left( x_1 \lor \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
Example

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

$$\varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4)$$
Example

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

$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$
Example

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

$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$
Example

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

$$\varphi = \left( x_1 \lor x_3 \lor x_4 \right) \land \left( x_1 \lor \neg x_2 \lor \neg x_3 \right) \land \left( \neg x_2 \lor \neg x_3 \lor x_4 \right)$$
Example

3SAT \leq_p CSAT

\varphi = (x_1 \lor \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)
Example

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

\[ \varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \]
Example

$3\text{SAT} \leq_{P} \text{CSAT}$

\[ \varphi = (x_1 \lor x_3 \lor x_4) \land (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \]
Lemma 24.4.

$SAT \leq_P 3SAT \leq_P CSAT$. 
THE END

... (for now)
24.2.2
Towards reducing CSAT to 3SAT
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 0 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 0 1 1 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 0 1 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 1 0 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 1 1 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 1 1 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0 1 1 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>z</th>
<th>x</th>
<th>y</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor y$</th>
<th>$\overline{z} \lor x \lor y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The formula $z = x \land y$ is satisfiable if and only if the corresponding clause in the 3SAT formula evaluates to 1.
Converting \( z = x \land y \) to 3SAT

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( z )</td>
<td>( x )</td>
<td>( y )</td>
<td>( z = x \land y )</td>
<td>( z \lor \overline{x} \lor y )</td>
<td>( \overline{z} \lor x \lor y )</td>
<td>( \overline{z} \lor x \lor \overline{y} )</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

0 0 0 \( = 1 \)
0 0 1 \( = 1 \)
0 1 0 \( = 1 \)
0 1 1 \( = 0 \)
1 0 0 \( = 0 \)
1 0 1 \( = 0 \)
1 1 0 \( = 1 \)
1 1 1 \( = 1 \)
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor \overline{y}$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>$z \lor \overline{x} \lor y$</th>
<th>$\overline{z} \lor x \lor \overline{y}$</th>
<th>$\overline{z} \lor \overline{x} \lor \overline{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting \( z = x \land y \) to 3SAT

<table>
<thead>
<tr>
<th>( z )</th>
<th>( x )</th>
<th>( y )</th>
<th>( z = x \land y )</th>
<th>( z \lor \overline{x} \lor \overline{y} )</th>
<th>( z \lor x \lor y )</th>
<th>( \overline{z} \lor x \lor \overline{y} )</th>
<th>( \overline{z} \lor \overline{x} \lor y )</th>
<th>( \overline{z} \lor \overline{x} \lor \overline{y} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
(z = x \land y) \\
\equiv \\
(z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)
\]
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Converting** $z = x \land y$ **to 3SAT**

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \bar{x} \lor \bar{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \land y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \land y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \bar{x} \lor \bar{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[
(z = x \land y) \\
\equiv \\
(z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y) \land (\bar{z} \lor x \lor \bar{y}) \land (\bar{z} \lor \bar{x} \lor y)
\]
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \bar{y}) = x$, we have that:
   1.1 $(\bar{z} \lor x \lor u) \land (\bar{z} \lor x \lor \bar{y}) = (\bar{z} \lor x)$
   1.2 $(\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor y) = (\bar{z} \lor y)$

2. Using the above two observation, we have that our formula
   $\psi \equiv (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y) \land (\bar{z} \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor \bar{x} \lor y)$

   is equivalent to $\psi \equiv (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x) \land (\bar{z} \lor y)$

Lemma 24.5.

$(z = x \land y) \equiv (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x) \land (\bar{z} \lor y)$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1.1 $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   1.2 $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula

   $\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)$

is equivalent to $\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)$

Lemma 24.5.

$(z = x \land y) \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)$
Converting $z = x \land y$ to $3SAT$

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1.1 $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   1.2 $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula

$$\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor x \lor \overline{y}) \land (\overline{z} \lor \overline{x} \lor y)$$

is equivalent to $\psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)$

Lemma 24.5.

$$(z = x \land y) \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y)$$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \neg y) = x$, we have that:
   1.1 $(\neg z \lor x \lor u) \land (\neg z \lor x \lor \neg y) = (\neg z \lor x)$
   1.2 $(\neg z \lor x \lor y) \land (\neg z \lor \neg x \lor y) = (\neg z \lor y)$

2. Using the above two observation, we have that our formula
   $\psi \equiv (z \lor \neg x \lor \neg y) \land (\neg z \lor x \lor y) \land (\neg z \lor x \land \neg y) \land (\neg z \lor x \lor \neg y)$
   is equivalent to $\psi \equiv (z \lor \neg x \lor \neg y) \land (\neg z \lor x) \land (\neg z \lor y)$

Lemma 24.5.

$(z = x \land y) \equiv (z \lor \neg x \lor \neg y) \land (\neg z \lor x) \land (\neg z \lor y)$
Converting $z = x \land y$ to 3SAT

Simplify further if you want to

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   
   1.1 $(\overline{z} \lor x \lor u) \land (\overline{z} \lor x \lor \overline{y}) = (\overline{z} \lor x)$
   
   1.2 $(\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) = (\overline{z} \lor y)$

2. Using the above two observation, we have that our formula

\[ \psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) \land (\overline{z} \lor \overline{x} \lor y) \land (\overline{z} \lor x \lor \overline{y}) \]

is equivalent to

\[ \psi \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y) \]

---

**Lemma 24.5.**

\[ (z = x \land y) \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y) \]
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$z \lor x \lor \bar{y}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$z \lor \bar{x} \lor y$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \bar{x} \lor \bar{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\bar{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Converting $z = x \lor y$ to 3SAT

<table>
<thead>
<tr>
<th>$z$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z = x \lor y$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$z \lor x \lor \overline{y}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$z \lor \overline{x} \lor y$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z \lor \overline{x} \lor \overline{y}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\overline{z} \lor x \lor y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$\left( z = x \lor y \right)$

$\equiv$

$\left( z \lor x \lor \overline{y} \right) \land \left( z \lor \overline{x} \lor y \right) \land \left( z \lor \overline{x} \lor \overline{y} \right) \land \left( \overline{z} \lor x \lor y \right)$
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

$$(z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y)$$

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1.1 $(z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) = z \lor \overline{y}$.
   1.2 $(z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{x}$

2. Using the above two observation, we have the following.

**Lemma 24.6.**

The formula $z = x \lor y$ is equivalent to the CNF formula

$$(z = x \lor y) \equiv (z \lor \overline{y}) \land (z \lor \overline{x}) \land (\overline{z} \lor x \lor y)$$
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

\[
(z = x \lor y) \equiv (z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y)
\]

1. Using that $(x \lor y) \land (x \lor \bar{y}) = x$, we have that:
   
   1.1 $(z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor \bar{y}) = z \lor \bar{y}$.
   1.2 $(z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) = z \lor \bar{x}$

2. Using the above two observation, we have the following.

**Lemma 24.6.**

The formula $z = x \lor y$ is equivalent to the CNF formula

\[
(z = x \lor y) \equiv (z \lor \bar{y}) \land (z \lor \bar{x}) \land (\bar{z} \lor x \lor y)
\]
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

$$ (z = x \lor y) \equiv (z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x \lor y) $$

1. Using that $(x \lor y) \land (x \lor \overline{y}) = x$, we have that:
   1.1 $(z \lor x \lor \overline{y}) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{y}$.
   1.2 $(z \lor \overline{x} \lor y) \land (z \lor \overline{x} \lor \overline{y}) = z \lor \overline{x}$

2. Using the above two observation, we have the following.

**Lemma 24.6.**

The formula $z = x \lor y$ is equivalent to the CNF formula

$$ (z = x \lor y) \equiv (z \lor \overline{y}) \land (z \lor \overline{x}) \land (\overline{z} \lor x \lor y) $$
Converting $z = x \lor y$ to 3SAT

Simplify further if you want to

$$(z = x \lor y) \equiv (z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x \lor y)$$

1. Using that $(x \lor y) \land (x \lor \bar{y}) = x$, we have that:
   
   1.1 $(z \lor x \lor \bar{y}) \land (z \lor \bar{x} \lor y) = z \lor \bar{y}$.
   
   1.2 $(z \lor \bar{x} \lor y) \land (z \lor \bar{x} \lor \bar{y}) = z \lor \bar{x}$

2. Using the above two observation, we have the following.

Lemma 24.6.

The formula $z = x \lor y$ is equivalent to the CNF formula

$$(z = x \lor y) \equiv (z \lor \bar{y}) \land (z \lor \bar{x}) \land (\bar{z} \lor x \lor y)$$
Converting $z = \bar{x}$ to CNF

**Lemma 24.7.**

\[ z = \bar{x} \iff (z \lor x) \land (\bar{z} \lor \bar{x}) \]
Summary of formulas we derived

**Lemma 24.8.**

The following identities hold:

1. \( z = \overline{x} \equiv (z \lor x) \land (\overline{z} \lor \overline{x}) \).
2. \( (z = x \lor y) \equiv (z \lor \overline{y}) \land (z \lor \overline{x}) \land (\overline{z} \lor x \land y) \)
3. \( (z = x \land y) \equiv (z \lor \overline{x} \lor \overline{y}) \land (\overline{z} \lor x) \land (\overline{z} \lor y) \)
THE END

...

(for now)
Reduction from CSAT to SAT
Converting a circuit into a CNF formula

Label the nodes

(A) Input circuit

(B) Label the nodes.
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>

From Lemma 24.8:

1. $z = \bar{x} \equiv (z \lor x) \land (\bar{z} \lor \bar{x})$
2. $(z = x \lor y) \equiv (z \lor \bar{y}) \land (z \lor \bar{x}) \land (\bar{z} \lor x \lor y)$
3. $(z = x \land y) \equiv (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x) \land (\bar{z} \lor y)$
Converting a circuit into a **CNF** formula

Convert each sub-formula to an equivalent CNF formula

<table>
<thead>
<tr>
<th>$x_k$ = $x_i \land x_j$</th>
<th>$(\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \land (x_k \lor \neg x_i \lor \neg x_j)$</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

| $x_d = 0$               | $\neg x_d$                                                                       |
| $x_a = 1$               | $x_a$                                                                            |

From **Lemma 24.8**:

1. $z = \bar{x}$ \quad \equiv \quad (z \lor x) \land (\bar{z} \lor \bar{x})$
2. $(z = x \lor y)$ \quad \equiv \quad (z \lor \bar{y}) \land (z \lor x) \land (\bar{z} \lor x \lor y)$
3. $(z = x \land y)$ \quad \equiv \quad (z \lor \bar{x} \lor \bar{y}) \land (\bar{z} \lor x) \land (\bar{z} \lor y)$
Converting a circuit into a **CNF** formula

Take the conjunction of all the **CNF** sub-formulas

\[
x_k \land (\neg x_k \lor x_i) \land (\neg x_k \lor x_j) \\
\land (x_k \lor \neg i \lor \neg j) \land (\neg j \lor x_g) \\
\land (\neg j \lor x_h) \land (x_j \lor \neg g \lor \neg h) \\
\land (x_i \lor x_f) \land (\neg i \lor \neg f) \\
\land (x_h \lor \neg d) \land (x_h \lor \neg e) \\
\land (\neg h \lor d \lor e) \land (g \lor \neg b) \\
\land (g \lor \neg c) \land (\neg g \lor b \lor c) \\
\land (\neg f \lor a) \land (\neg f \lor b) \\
\land (f \lor \neg a \lor \neg b) \land (\neg d) \land a
\]

We got a **CNF** formula that is satisfiable if and only if the original circuit is satisfiable.
Correctness of Reduction

Need to show circuit $C$ is satisfiable if and only if $\varphi_C$ 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 $C$
The result

**Lemma 24.9.**

\[ CSAT \leq_p SAT \leq_p 3SAT. \]

**Theorem 24.10.**

*CSAT* is NP-Complete.
The result

**Lemma 24.9.**
\[ \text{CSAT} \leq_p \text{SAT} \leq_p \text{3SAT}. \]

**Theorem 24.10.**
\[ \text{CSAT} \text{ is NP-Complete.} \]
THE END
...
(for now)
24.3

NP-Completeness of Graph Coloring
24.3.1

The coloring problem
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?
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 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).
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).
THE END

...(for now)
24.3.2
Problems related to graph coloring
Register allocation during compilation

1. When a compiler generates the assembly/VM code it needs to allocation registers to values being handled.
2. Need to make sure registers are not in conflict.
3. Build a conflict graph.
4. Color the conflict graph.
5. Every color is a register.
6. If not enough registers, then use memory/stack to store values.
7. CISC v.s. RISC.
Register allocation during compilation

1. When a compiler generates the assembly/VM code it needs to allocation registers to values being handled.
2. Need to make sure registers are not in conflict.
3. Build a conflict graph.
4. Color the conflict graph.
5. Every color is a register.
6. If not enough registers, then use memory/stack to store values.
7. CISC v.s. RISC.
Graph Coloring and Register Allocation

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.
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.
Frequency Assignments in Cellular Networks

1. 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

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

3. Can reduce to $k$-coloring by creating interference/conflict graph on towers.
Frequency Assignments in Cellular Networks

1. 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 \([a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]\)
   - Each cell phone tower (simplifying) gets one band
   - Constraint: nearby towers cannot be assigned same band, otherwise signals will interfere

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

3. Can reduce to \(k\)-coloring by creating interference/conflict graph on towers.
Frequency Assignments in Cellular Networks

1. 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

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

3. Can reduce to \(k\)-coloring by creating interference/conflict graph on towers.
THE END

...

(for now)
24.3.3

Showing NP-Completeness of 3

COLORING
24.3.3.1
The variable assignment gadget
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. \( \phi \): 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 \( 3\text{SAT} \) formula (i.e., \( 3\text{CNF} \) 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-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)
Reduction idea

1. Let \( \varphi \) be a \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 $3\text{SAT}$ formula (i.e., $3\text{CNF}$ 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)
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)
Assignment encoding using 3-coloring
THE END

... (for now)
24.3.3.2
The clause gadget
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).

(A) Yes.
(B) No.
3 color this gadget II

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).

(A) Yes.
(B) No.
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:
**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:

```
 a
 b
 c
 a \lor b
 a \lor b \lor c
```
**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
Claim 24.1.

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
- FFT
- FTF
- FTT
- TFF
- TFT
- TTF
- TTT
Example 24.2.

\[ \varphi = (u \lor \neg v \lor w) \land (v \lor x \lor \neg y) \]
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 \( \overline{v}_i \) 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 \( \overline{v}_i \) 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 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!
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

\((a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\)
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

\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]
THE END

...  

(for now)
24.4

Proof of Cook-Levin Theorem
24.4.1
Statement and sketch of idea for the proof
Cook-Levin Theorem

**Theorem 24.1 (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.
Cook-Levin Theorem

**Theorem 24.1 (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.
The plot against SAT
High-level plan to proving the Cook-Levin theorem

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

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

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

---

**Input:** $M, x, p$

**Question:** Does $M$ stops on input $x$ after $p(|x|)$ steps?

Describe a reduction $R$ that computes from $M, x, p$ a SAT formula $\varphi$.

- $R$ takes as input a string $x$ and outputs a SAT formula $\varphi$
- $R$ runs in time polynomial in $|x|, |M|$
- $x \in L$ if and only if $\varphi$ is satisfiable
The plot against SAT

High-level plan to proving the Cook-Levin theorem

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

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

---

**Input:** \( M, x, p. \)

**Question:** Does \( M \) stops on input \( x \) after \( p(|x|) \) steps?

Describe a reduction \( R \) that computes from \( M, x, p \) a SAT formula \( \varphi \).

- \( R \) takes as input a string \( x \) and outputs a SAT formula \( \varphi \)
- \( R \) runs in time polynomial in \( |x|, |M| \)
- \( x \in L \) if and only if \( \varphi \) is satisfiable
The plot against SAT
High-level plan to proving the Cook-Levin theorem

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

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

---

Input: $M, x, p$.
Question: Does $M$ stops on input $x$ after $p(|x|)$ steps?

Describe a reduction $R$ that computes from $M, x, p$ a SAT formula $\varphi$.

- $R$ takes as input a string $x$ and outputs a SAT formula $\varphi$
- $R$ runs in time polynomial in $|x|, |M|$
- $x \in L$ if and only if $\varphi$ is satisfiable
The plot against SAT continued

\[ \langle x, M, p \rangle \rightarrow R \rightarrow \varphi \]

poly-time computable

\( \varphi \) is satisfiable if and only if \( x \in L \)
\( \varphi \) is satisfiable if and only if nondeterministic \( M \) accepts \( x \) in \( p(|x|) \) steps

BIG IDEA

- \( \varphi \) will express “\( M \) on input \( x \) accepts in \( p(|x|) \) steps”
- \( \varphi \) will encode a computation history of \( M \) on \( x \)

\( \varphi \): CNF formula s.t if we have a satisfying assignment to it \( \implies \) accepting computation of \( M \) on \( x \) down to the last details (where the head is, what transition is chosen, what the tape contents are, at each step, etc).
The plot against SAT continued

\[ \langle x, M, p \rangle \xrightarrow{\text{poly-time computable}} \varphi \]

\( \varphi \) is satisfiable if and only if \( x \in L \)
\( \varphi \) is satisfiable if and only if nondeterministic \( M \) accepts \( x \) in \( p(|x|) \) steps

BIG IDEA

▶ \( \varphi \) will express “\( M \) on input \( x \) accepts in \( p(|x|) \) steps”
▶ \( \varphi \) will encode a computation history of \( M \) on \( x \)

\( \varphi \): CNF formula s.t if we have a satisfying assignment to it \( \implies \) accepting computation of \( M \) on \( x \) down to the last details (where the head is, what transition is chosen, what the tape contents are, at each step, etc).
The plot against SAT continued

\[ \langle x, M, p \rangle \rightarrow^R \varphi \]

\( \varphi \) is satisfiable if and only if \( x \in L \)
\( \varphi \) is satisfiable if and only if nondeterministic \( M \) accepts \( x \) in \( p(|x|) \) steps

BIG IDEA

- \( \varphi \) will express “\( M \) on input \( x \) accepts in \( p(|x|) \) steps”
- \( \varphi \) will encode a computation history of \( M \) on \( x \)

\( \varphi \): CNF formula s.t if we have a satisfying assignment to it \( \iff \) accepting computation of \( M \) on \( x \) down to the last details (where the head is, what transition is chosen, what the tape contents are, at each step, etc).
The plot against SAT continued

\( \langle x, M, p \rangle \xrightarrow{R} \varphi \) poly-time computable

\( \varphi \) is satisfiable if and only if \( x \in L \)

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

**BIG IDEA**

- \( \varphi \) will express “\( M \) on input \( x \) accepts in \( p(|x|) \) steps”
- \( \varphi \) will encode a computation history of \( M \) on \( x \)

\( \varphi \): CNF formula s.t if we have a satisfying assignment to it \( \Rightarrow \) accepting computation of \( M \) on \( x \) down to the last details (where the head is, what transition is chosen, what the tape contents are, at each step, etc).
The Matrix Executions

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.
Variables of $\varphi$

Four types of variables to describe computation of $M$ on $x$

- $T(b, h, i)$: tape cell at position $h$ holds symbol $b$ at time $i$.
  For $h = 1, \ldots, p(|x|)$, $b \in \Gamma$, $i = 0, \ldots, p(|x|)$.

- $H(h, i)$: read/write head is at position $h$ at time $i$.
  For $h = 1, \ldots, p(|x|)$, and $i = 0, \ldots, p(|x|)$.

- $S(q, i)$ state of $M$ is $q$ at time $i$.
  For all $q \in Q$ and $i = 0, \ldots, p(|x|)$.

- $I(j, i)$ 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 $O(p(|x|)^2|M|^2)$.
**Notation**

Some abbreviations for ease of notation

\[ \bigwedge_{k=1}^{m} x_k \] means \( x_1 \land x_2 \land \ldots \land x_m \)

\[ \bigvee_{k=1}^{m} x_k \] 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 **CNF** form

**CNF** formula showing making sure that at most one variable is assigned value 1:

\[
\bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j})
\]

Making sure that one of the variables is true: \( \bigvee_{i=1}^{k} x_i \).

\[ \bigoplus(x_1, x_2, \ldots, x_k) = \bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j}) \land (x_1 \lor x_2 \lor \cdots \lor x_k). \]
Notation

Some abbreviations for ease of notation

$\bigwedge_{k=1}^{m} x_k$ means $x_1 \land x_2 \land \ldots \land x_m$

$\bigvee_{k=1}^{m} x_k$ 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 CNF form

CNF formula showing making sure that at most one variable is assigned value 1:

$$\bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j})$$

Making sure that one of the variables is true: $\bigvee_{i=1}^{k} x_i$.

$$\bigoplus(x_1, x_2, \ldots, x_k) = \bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j}) \bigwedge (x_1 \lor x_2 \lor \cdots \lor x_k).$$
Notation

Some abbreviations for ease of notation

$\bigwedge_{k=1}^{m} x_k$ means $x_1 \land x_2 \land \ldots \land x_m$

$\bigvee_{k=1}^{m} x_k$ 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 CNF form

CNF formula showing making sure that at most one variable is assigned value 1:

$$\bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j})$$

Making sure that one of the variables is true: $\bigvee_{i=1}^{k} x_i$.

$$\bigoplus(x_1, x_2, \ldots, x_k) = \bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j}) \land (x_1 \lor x_2 \lor \cdots \lor x_k).$$
Notation

Some abbreviations for ease of notation

\( \bigwedge_{k=1}^{m} x_k \) means \( x_1 \land x_2 \land \ldots \land x_m \)

\( \bigvee_{k=1}^{m} x_k \) 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 CNF form

**CNF** formula showing making sure that at most one variable is assigned value 1:

\[
\bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j})
\]

Making sure that one of the variables is true: \( \bigvee_{i=1}^{k} x_i \).

\[
\bigoplus(x_1, x_2, \ldots, x_k) = \bigwedge_{1 \leq i < j \leq k} (\overline{x_i} \lor \overline{x_j}) \bigwedge (x_1 \lor x_2 \lor \ldots \lor x_k).
\]
Clauses of $\varphi$

$\varphi$ is the conjunction of 8 clause groups:

$$\varphi = \bigwedge_{i=1}^{12} \varphi_i$$

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

**Property:** $\varphi$ is satisfied $\iff$ there is an execution of $M$ on $x$ that accepts the language in $p(|x|)$ time.
THE END

...

(for now)
24.4.2
The consistency of execution
The variables of $\varphi$

**Variables:**

$\langle q_j, b_j, q'_j, b'_j, d_j \rangle$: $j$th instruction of $M$

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
$\varphi_1$: The input is encoded correctly

$\varphi_1$ 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 $x = x_1x_2 \ldots x_n$

$$\varphi_1 = S(q_0, 0)$$

$$\bigwedge_{h=1}^{n} T(x_h, h, 0)$$

// at time 0 cells 1 to n have value $x_1$ to $x_n$

$$\bigwedge_{h=n+1}^{p(n)} T(\perp, h, 0)$$

// all remaining cells are blank

$$\bigwedge H(1, 0)$$

// The head is at time 0 at start of tape
φ₂: $M$ is in exactly one state at any point in time

φ₂ asserts $M$ in exactly one state at any time $i$:

$$φ₂ = \bigwedge_{i=0}^{p(|x|)} \left( \bigoplus (S(q₀, i), S(q₁, i), \ldots, S(q|Q|, i)) \right)$$

**Variables:**

$q_j, b_j, q_j', b_j', d_j$: $j$th instruction of $M$

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
**\( \varphi_3 \): Each tape cell holds a unique symbol at any time**

\( \varphi_3 \) 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 \( i \)

**Variables:**

\[
\langle q_j, b_j, q'_j, b'_j, d_j \rangle: \text{ } j\text{-th instruction of } M
\]

\( I(j, i) \): Instruction \( j \) was issued at time \( i \).

\( H(h, i) \): The head is at location \( h \) at time \( i \).

\( T(c, h, i) \): The tape at location \( h \) at time \( i \) stored the character \( c \).
\( \varphi_4: \) tape head of \( M \) is in exactly one position at any time \( i \)

\( \varphi_4 \) asserts that the read/write head of \( M \) is in exactly one position at any time \( i \)

\[
\varphi_4 = \bigwedge_{i=0}^{p(|x|)} (\bigoplus (H(1, i), H(2, i), \ldots, H(p(|x|), i)))
\]

**Variables:**

\[ \langle q_j, b_j, q'_j, b'_j, d_j \rangle: \text{\( j \)th instruction of \( M \)} \]

\[ I(j, i): \text{Instruction \( j \) was issued at time \( i \).} \]

\[ H(h, i): \text{The head is at location \( h \) at time \( i \).} \]

\[ T(c, h, i): \text{The tape at location \( h \) at time \( i \) stored the character \( c \).} \]
**φ₅: M accepts the input**

φ₅ asserts that M accepts

- Let qa be unique accept state of M
- Without loss of generality assume M runs all \( p(|x|) \) steps

\[
φ₅ = S(q_a, p(|x|))
\]

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

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

\[
φ₅ = \bigvee_{i=1}^{p(|x|)} S(q_a, i)
\]

which means M enters accepts state at some time.
\( \varphi_6: \ M \) executes a unique instruction at each time

\( \varphi_6 \) asserts that \( M \) executes a unique instruction at each time

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

where \( m \) is max instruction number.

**Variables:**

\( \langle q_j, b_j, q'_j, b'_j, d_j \rangle \): \( j \)th instruction of \( M \)

\( I(j, i) \): Instruction \( j \) was issued at time \( i \).

\( H(h, i) \): The head is at location \( h \) at time \( i \).

\( T(c, h, i) \): The tape at location \( h \) at time \( i \) stored the character \( c \).
\( \varphi_7: \) Tape changes only because of the head writing something

\( \varphi_7 \) 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”

\[
\varphi_7 = \land_i \land_h \land_{b \neq c} \left( H(h, i) \Rightarrow T(b, h, i) \land T(c, h, i + 1) \right)
\]

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

\[
\varphi_7 = \land_i \land_h \land_{b \neq c} \left( H(h, i) \lor \neg T(b, h, i) \lor \neg T(c, h, i + 1) \right)
\]
$\varphi_8$: Transitions are done from correct states

$j$th instruction of $M$: $< q_j, b_j, q'_j, b'_j, d_j >$

$$\varphi_8 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q_j, i))$$

If instruction $j$ is executed at time $i$ then state at time $i$ must be $q_j$.

---

**Variables:**

$\langle q_j, b_j, q'_j, b'_j, d_j \rangle$: $j$th instruction of $M$

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
$\varphi_9$: Transitions are done into correct state

$j$th instruction of $M$: $< q_j, b_j, q'_j, b'_j, d_j >$

$$\varphi_9 = \bigwedge_i \bigwedge_j (I(j, i) \Rightarrow S(q'_j, i + 1))$$

If instruction $j$ was performed at time $i$, then state at time $i + 1$ must be $q'_j$.

Variables:

$\langle q_j, b_j, q'_j, b'_j, d_j \rangle$: $j$th instruction of $M$

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
\(\varphi_{10}: \) The character written on tape that triggered an instruction, is the correct one

\[
\varphi_{10} = \bigwedge_i \bigwedge_h \bigwedge_j [(l(j, i) \land H(h, i)) \Rightarrow T(b_j, h, i)]
\]

If instruction \(j\) was executed at time \(i\) and head was at position \(h\), then cell \(h\) has the symbol needed to issue instruction \(j\) is written under the head location on the tape.

**Variables:**

\(<q_j, b_j, q'_j, b'_j, d_j>: j\)th instruction of \(M\)

\(l(j, i): \) Instruction \(j\) was issued at time \(i\).

\(H(h, i): \) The head is at location \(h\) at time \(i\).

\(T(c, h, i): \) The tape at location \(h\) at time \(i\) stored the character \(c\).
**$\varphi_{11}$**: The correct symbol was written to the tape at time $i$

\[
\varphi_{11} = \bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \land H(h, i)) \Rightarrow T(b'_j, h, i + 1)]
\]

If instruction $j$ was executed time $i$ with head at $h$, then at next time step symbol $b'_j$ was written in position $h$

**Variables:**

\[
\langle q_j, b_j, q'_j, b'_j, d_j \rangle: j\text{th instruction of } M
\]

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
$\varphi_{12}$: Head was moved in the right direction at time $i$

$$\varphi_{12} = \bigwedge_i \bigwedge_j \bigwedge_h [(I(j, i) \land H(h, i)) \Rightarrow H(h + d_j, i + 1)]$$

The head is moved properly according to instr $j$.

**Variables:**

$\langle q_j, b_j, q'_j, b'_j, d_j \rangle$: $j$th instruction of $M$

$I(j, i)$: Instruction $j$ was issued at time $i$.

$H(h, i)$: The head is at location $h$ at time $i$.

$T(c, h, i)$: The tape at location $h$ at time $i$ stored the character $c$. 
THE END
...
(for now)
24.4.3

Proof of correctness
Proof of Correctness

(Sketch)

- Given $M$, $x$, poly-time algorithm to construct $\varphi$
- if $\varphi$ 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 $\varphi$. Simply assign the variables according to the state of $M$ and cells at each time $i$.

Thus $M$ accepts $x$ if and only if $\varphi$ is satisfiable.
THE END

... 

(for now)
24.5

NP-Complete problems to know and remember
# List of NP-Complete Problems to Remember

<table>
<thead>
<tr>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>SAT</strong></td>
</tr>
<tr>
<td>2. <strong>3SAT</strong></td>
</tr>
<tr>
<td>3. <strong>CircuitSAT</strong></td>
</tr>
<tr>
<td>4. <strong>Independent Set</strong></td>
</tr>
<tr>
<td>5. <strong>Clique</strong></td>
</tr>
<tr>
<td>6. <strong>Vertex Cover</strong></td>
</tr>
<tr>
<td>7. <strong>Hamilton Cycle</strong> and <strong>Hamilton Path</strong> in both directed and undirected graphs</td>
</tr>
<tr>
<td>8. <strong>3Color</strong> and <strong>Color</strong></td>
</tr>
</tbody>
</table>