Recall:

\[ L' \in NP \]
\[ L \ NP-complete \] \[ \Rightarrow \]
\[ L \leq_p L' \]

At this moment, we know that CIRCUIT-SAT is NP-complete.

We will prove that 3SAT is NP-Complete.

It is sufficient to show:

* 3SAT \( \in \) NP: for a given truth-assignment of the variables, we can verify in polynomial time if the Boolean formula is true.

* CIRCUIT-SAT \( \leq_p \) 3SAT.
We need a function $f$ such that

1. $f: \text{Boolean circuit } B \rightarrow \text{Boolean formula } \varphi$
   as on page 210

2. $\exists$ truth-values for the unknown input gates such that $B$'s output is true
   $\iff \exists$ truth-values for the variables such that $\varphi$ is true

3. $\varphi = f(B)$ can be computed in time that is polynomial in the length of $B$.

Consider a Boolean circuit $B$.
- one variable for each gate.
- describe the effect of each gate using a few clauses.
- connect all clauses by $\land$'s.
known input gates:

\[ \text{TRUE} \iff \text{clause: } x \]

\[ \text{FALSE} \iff \text{clause: } \neg x \]

unknown input gates:

\[ ? \iff \text{clause: } x \lor \neg x \]

NOT-gates:

\[ \text{NOT} \]

\[ y \]

\[ \iff \text{clauses: } (y \Rightarrow \neg x) \land (\neg y \Rightarrow x) \]

\[ \text{Same as} \]

\[ (\neg y \lor \neg x) \land (y \lor x) \]
OR-gates:

\[ (x \Rightarrow (y \lor z)) \land \\
( (y \lor z) \Rightarrow x ) \]

Same as

\[ (\neg x \lor y \lor z) \land (\neg y \lor x) \land (\neg z \lor x) \]

AND-gates:

\[ (x \Rightarrow (y \land z)) \land ( (y \land z) \Rightarrow x ) \]

Same as

\[ (\neg x \lor y) \land (\neg x \lor z) \land (\neg y \lor z \lor x) \]

output-gate:

\[ clause: \ x \]
Let \( \varphi \) be the conjunction of all these clauses.

By construction:

\[
\text{B \in \text{CIRCUIT-SAT}} \iff f(B) = \varphi \in \text{3SAT}.
\]

Size of \( \varphi \):

* number of variables in \( \varphi \) = number of gates in \( B \)
* each clause has \( \leq 3 \) literals
* number of clauses \( \leq 3 \times \) number of gates in \( B \)

\[\therefore \text{size of } \varphi = O(\text{size of } B) : \text{polynomial}\]

\( \varphi \) can be computed in time that is polynomial in the size of \( B \).

Conclusion: 3SAT is NP-complete.
3SAT \text{ NP-complete}

3SAT \leq_p \text{ INDEP-SET (page 193)} \text{ \{ imply \}} \text{ INDEP-SET } \in \text{ NP (exercise)} \\text{ NP-complete.}

\text{ CLIQUE } \in \text{ NP (exercise)} \text{ \{ imply \}} \text{ CLIQUE } \text{ NP-complete}

\text{ INDEP-SET } \leq_p \text{ CLIQUE (page 186)} \text{ \{ imply \}} \text{ INDEP-SET } \text{ NP-complete}

\text{ VERTEX-COVER } \in \text{ NP (exercise)} \text{ \{ imply \}} \text{ VERTEX-COVER } \text{ NP-complete}

\text{ CLIQUE } \leq_p \text{ VERTEX-COVER (page 188)} \text{ \{ imply \}} \text{ CLIQUE } \text{ NP-complete.}
0/1 integer programming

Input: Linear inequalities such as

\[
\begin{align*}
3y_1 - 2y_2 - 5y_3 &\leq -2 \\
-y_1 + 7y_2 - 2y_3 &\leq -3
\end{align*}
\]

Question: Are there binary values for the variables such that all inequalities hold?

For the example: \( y_1 = 1, y_2 = 0, y_3 = 1 \).

We can write the example as

\[
\begin{pmatrix}
3 & -2 & -5 \\
-1 & 7 & -2
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
y_3
\end{pmatrix}
\leq
\begin{pmatrix}
-2 \\
-3
\end{pmatrix}
\]

component-wise

\[ Ay \leq c \]

- \( A \): \( m \times n \) matrix, \( m = \#\) inequalities, \( n = \#\) variables
- \( y \): binary vector of length \( n \) : \( y \in \{0,1\}^n \)
- \( c \): vector of length \( m \)
0/1-IP =

\{(A,c) : A is an integer m \times n matrix, c is an integer column m-vector, \exists y \in \{0,1\}^n : Ay \leq c\}.

Exercise: Show that $0/1$-IP $\in$ NP.

We will show: 3SAT $\leq_p$ 0/1-IP.

Since 3SAT is NP-complete, $0/1$-IP is NP-complete.

We need a function f:

1. $f$: input $\phi$ for 3SAT $\rightarrow$ input $(A,c)$ for 0/1-IP

2. $\phi$ satisfiable $\iff \exists$ binary vector $y : Ay \leq c$

3. Time to compute $(A,c)$ is polynomial in the length of $\phi$. 
\[ \varphi = (x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \]

\[ C_1 \quad C_2 \]

We are going to write "\( \varphi \) is satisfiable" as a system of linear inequalities.

We will use variables \( y_1, \ldots, y_6 \):

**Correspondence:**

\[
\begin{align*}
x_1 & \iff y_1 \\
\neg x_1 & \iff y_2 \\
x_2 & \iff y_3 \\
\neg x_2 & \iff y_4 \\
x_3 & \iff y_5 \\
\neg x_3 & \iff y_6
\end{align*}
\]

Remember: \( x_1, x_2, x_3 \): Boolean variables; true or false
\( y_1, \ldots, y_6 \): binary variables; 0 or 1
\( q \) is satisfiable \( \iff \)

1. for \( i=1,2,3 \): we can give \( x_i \) a truth-value
2. for \( j=1,2 \): clause \( C_j \) is true.

\( x_1 \): has a truth-value
\( \iff \) \( y_1 + y_2 = 1 \)
\( \iff \) \( y_1 + y_2 \leq 1 \) and \( y_1 + y_2 \geq 1 \)
\( \iff \)
\( y_1 + y_2 \leq 1 \)
\( -y_1 - y_2 \leq -1 \)

\( x_2 \): has a truth-value \( \iff \)
\( y_3 + y_4 \leq 1 \)
\( -y_3 - y_4 \leq -1 \)

\( x_3 \): has a truth-value \( \iff \)
\( y_5 + y_6 \leq 1 \)
\( -y_5 - y_6 \leq -1 \)
\( C_1 = x_1 \lor \neg x_2 \lor x_3 \) is true
\[\iff\] at least one of \( x_1, \neg x_2, x_3 \) is true
\[\iff\] \( y_1 + y_4 + y_5 \geq 1 \)
\[\iff\] \( -y_1 - y_4 - y_5 \leq -1 \)

\( C_2 = \neg x_1 \lor x_2 \lor \neg x_3 \) is true
\[\iff\] \( -y_2 - y_3 - y_6 \leq -1 \)

Thus: \( y \) is mapped to

\[
A = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
-1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 0 & 0 & -1 & 0
\end{pmatrix},
\]

\[
C = \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix}
\]
In general:

Boolean variables $x_1, \ldots, x_n$

$\phi = C_1 \land \cdots \land C_m$, $C_i = l_i^1 \lor l_i^2 \lor l_i^3$,

each $l_i^j$ is either $x_k$ or $\neg x_k$ for some $k$.

Our linear inequalities will use binary variables $y_1, \ldots, y_{2n}$.

$A$ will be a $(2n+m) \times 2n$ matrix.

$c$ will be a $(2n+m)$-vector.

\[
C = \begin{bmatrix}
\vdots \\
-1 \\
\vdots \\
-1 \\
\vdots \\
-1 \\
\vdots \\
-1 \\
\end{bmatrix}_{2n \times m}
\]
for $i = 1, \ldots, n$:
row $2i-1$ of $A$:
\[
\begin{array}{c}
2(i-1) \\
\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]
row $2i$ of $A$:
\[
\begin{array}{c}
2(n-i) \\
\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

for $i = 1, \ldots, m$:
what is row $2n+i$ of $A$:

for $j = 1, 2, 3$:
if $l^j_i = x_k$:
entry in column $2k-1$ is $-1$
if $l^j_i = -x_k$:
entry in column $2k$ is $-1$
all other entries are $0$

This defines the function $f : \varphi \rightarrow (A, c)$.

By construction:
for \( \varphi \) satisfiable \( \iff \exists y \in \{0,1\}^{2n} : Ay \leq c \).

Time to construct $(A, c)$:
\[
O((n+m)n) = O((n+m)^2) = O((\text{length of } \varphi)^2) \text{ polynomial}
\]