→ Ruorata cycle is NP-complete
(HAMILTON).

→ Circuit SAT is NP-complete
(Mother of all NP-completeness results).

Every problem in NP \leq_p Circuit SAT

→ Circuit SAT \leq_p 3SAT
RUORATA CYCLE (directed)
Input: A directed graph $G = (V,E)$
Solution: A cycle passing through all nodes exactly once

RUORATA CYCLE $\in$ NP

Goal: RUORATA CYCLE is NP-complete

Thm: 3SAT $\leq_P$ RUORATA CYCLE
$3SAT \leq_p RUDRATA\ CYCLE$

$3SAT$ instance:
\[
(x \lor y \lor z) \land (\neg x \lor z \lor w) \land (\neg x \lor y \lor \neg w)
\]

Reduction:

Input: Directed Graph $G=(V,E)$

Algorithm:

\[
\exists\ \text{a satisfying assignment to } \phi \implies \exists\ \text{a Rudrata Cycle in } G
\]

\[
\exists\ \text{a Rudrata cycle in } G \implies \exists\ \text{a satisfying assignment in } \phi
\]
\[ x \in \{0, 1\} \]

\[ x = 0 \iff \text{Right to left} \]

\[ -1 \iff \text{left to right}\]

(Traverse the graph without skipping/repeating any vertex \( u \))
Every boolean assignment

$3\text{SAT} \rightarrow \text{Rudrata Cycle}$

$\chi_1$

$\chi_2$

$\chi_n$
$C = (x \lor \overline{y} \lor \overline{z})$

$\forall \text{ clause in 3SAT}
$
add a vertex, and connect it only after

cycle covers $C$

$x \in L \rightarrow R$

$y \in R \rightarrow L$

$z \in L \rightarrow R$
CIRCUIT SAT

INPUT: 1) Circuit with AND/OR/NOT gates
   2) n inputs

SOLUTION: An assignment so that output = 1.
Circuit SAT in NP-complete:

Every problem in NP $\leq_p$ Circuit SAT

Examples: Factorization $\leq_p$
FACTORIZATION
Input: An n bit number \( N \)
Sol: \( p, q > 1 \) and \( p \cdot q = N \)

CIRCUIT SAT
Input: Circuit \( C \)
Sol: \( x \) s.t. \( C(x) = 1 \)

Reduction Alg
\( N \)
\( \rightarrow \)
\( 1011 \)

CIRCUIT SAT
Alg
\( C \)
\( \rightarrow \)
\( 111111 \)

Verification circuit for factorisation with input \( N \text{(fixed)} \) (hard coded)
Polynomial

Verification Algorithm for Factorization

polytime Circuit

Verification Circuit

Graph G

Cycle C
Same proof applies to every problem in NP

\[
\implies \quad \text{Every problem in NP} \leq_p \text{Circuit SAT}. 
\]

\[
\implies \text{Circuit SAT is NP-complete.}
\]
CIRCUIT SAT

Input: A circuit C

Solution: An assignment x, s.t.

\[ C(x) = 1 \]

3SAT

Input: A 3SAT formula

\[(x_1 \lor x_2 \lor x_3) \land (\neg x_3 \lor \neg x_5 \lor x_4) \land \ldots\]

Solution: A satisfying assignment

\[
\begin{align*}
\text{1) For each wire } w_i, & \text{ introduce a variable in 3SAT} \\
\{w_1, \ldots, w_7\} \\
\text{2) For each gate introduce clauses} \\
\text{Example: we want} \\
w_6 = w_1 \lor w_2 \\
\end{align*}
\]

Simulate this using clauses
Observation: Every constraint on 3-variables can be simulated via clauses of form \((x \lor y \lor z)\).

Example: Suppose we want a constraint \(x = y \lor z\), then

\[
\begin{align*}
(x \lor \neg y \lor \neg z) & , \\
(\neg x \lor y \lor \neg z) & , \\
(\neg x \lor \neg y \lor z) & , \\
(\neg x \lor y \lor z) & .
\end{align*}
\]

4 clauses simulate \(x = y \lor z\).
To express a constraint using 3SAT clauses

Example: \( x = y \lor z \)

Forbidden assignments

\[
\begin{array}{c c c}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 0 & 0 \\
\end{array}
\quad \rightarrow \quad \begin{array}{c}
(x \lor \overline{y} \lor \overline{z}) \\
(\overline{x} \lor y \lor \overline{z}) \\
(\overline{x} \lor \overline{y} \lor z) \\
(\overline{x} \lor y \lor z) \\
\end{array}
\]