Steps to Proving Problem $R$ is NP-complete

- **Show $R$ is NP-hard:**
  - State: Reduction is from NP-hard Problem $L$.
  - Show what the map is.
  - Argue that the map is polynomial time.
  - Argue correctness: two directions Yes for $L$ implies Yes for $R$ and vice versa.

- **Show $R$ is in NP**
  - State what hint is and why it works.
  - Argue that it is polynomial-time to check.

Problems we already know are NP-complete

- Satisfiability
- Independent-Set
- Clique
- Vertex Cover

There are 1000’s of practical problems that are NP-complete, e.g. scheduling, optimal VLSI layout etc.

A particularly useful problem for proving NP-completeness

- 3-SAT: Given a CNF formula $F$ having precisely 3 variables per clause (i.e., in 3-CNF), is $F$ satisfiable?

- **Claim:** 3-SAT is NP-complete

- **Proof:**
  - $3$-SAT $\in$ NP
    - Hint is a satisfying assignment
    - Just like Satisfiability it is polynomial-time to check the hint

Satisfiability $\leq^P$ 3-SAT

- **Reduction:**
  - mapping CNF formula $F$ to another CNF formula $G$ that has precisely 3 variables per clause.
  - $G$ has one or more clauses for each clause of $F$.
  - $G$ will have extra variables that don’t appear in $F$.
  - For each clause $C$ of $F$ there will be a different set of variables that are used only in the clauses of $G$ that correspond to $C$.

Satisfiability $\leq^P$ 3-SAT

- **Goal:**
  - An assignment $A$ to the original variables makes clause $C$ true in $F$ if
    - there is an assignment to the extra variables that together with the assignment $A$ will make all new clauses corresponding to $C$ true.

  Define the reduction clause-by-clause

  - We’ll use variable names $z_i$ to denote the extra variables related to a single clause $C$ to simplify notation.
  - In reality, two different original clauses will not share $z_i$. 
**Satisfiability \(\leq_p \text{3-SAT}**

For each clause \(C\) in \(F\):

- If \(C\) has 3 variables:
  - Put \(C\) in \(G\) as is

- If \(C\) has 2 variables, e.g. \(C = (x_1 \vee \neg x_2)\)
  - Use a new variable \(z\) and put two clauses in \(G\):
    
    \[
    (x_1 \vee \neg x_2 \vee z) \land (x_1 \vee \neg x_2 \vee \neg z)
    \]
  - If original \(C\) is true under assignment \(A\) then both new clauses will be true under \(A\)
  - If new clauses are both true under some assignment \(B\) then the value of \(z\) doesn't help in one of the two clauses so \(C\) must be true under \(B\)

- If \(C\) has 1 variable: e.g. \(C = x_1\)
  - Use two new variables \(z_1, z_2\) and put 4 new clauses in \(G\):
    
    \[
    (x_1 \vee \neg z_1 \vee \neg z_2) \land (x_1 \vee z_1 \vee \neg z_2) \land (x_1 \vee \neg z_1 \vee z_2) \land (x_1 \vee z_1 \vee z_2)
    \]
  - If original \(C\) is true under assignment \(A\) then all new clauses will be true under \(A\)
  - If new clauses are all true under some assignment \(B\) then the values of \(z_1\) and \(z_2\) doesn't help in one of the 4 clauses so \(C\) must be true under \(B\)

- If \(C\) has \(k \geq 4\) variables: e.g. \(C = (x_1 \vee ... \vee x_k)\)
  - Use \(k-3\) new variables \(z_2, z_3, ..., z_{k-2}\) and put \(k-2\) new clauses in \(G\):
    
    \[
    (x_1 \vee x_2 \vee z_2) \land (\neg z_2 \vee x_3 \vee z_3) \land (\neg z_3 \vee x_4 \vee z_4) \land ...
    \]
    
    \[
    \land (\neg z_{k-3} \vee x_{k-2} \vee z_{k-2}) \land (\neg z_{k-2} \vee x_{k-1} \vee x_k)
    \]
  - If original \(C\) is true under assignment \(A\) then some \(x_i\) is true for \(i \leq k\). By setting \(z_j\) true for all \(j < i\) and false for all \(j \geq i\), we can extend \(A\) to make all new clauses true.
  - If new clauses are all true under some assignment \(B\) then some \(x_i\) must be true for \(i \leq k\) because \(z_2 \land (\neg z_2 \vee z_3) \land ... \land (\neg z_{k-3} \vee z_{k-2}) \land \neg z_{k-2}\) is not satisfiable


**Graph Colorability**

- Defn: Given a graph \(G = (V,E)\), and an integer \(k\), a \(k\)-coloring of \(G\) is:
  - an assignment of up to \(k\) different colors to the vertices of \(G\) so that the endpoints of each edge have different colors.

- **3-Color**: Given a graph \(G = (V,E)\), does \(G\) have a 3-coloring?

- **Claim**: 3-Color is NP-complete

- **Proof**
  - Hint is an assignment of red,green,blue to the vertices of \(G\)
  - Easy to check that each edge is colored correctly

**3-SAT \(\leq_p \text{3-Color}**

Reduction:

We want to map a 3-CNF formula \(F\) to a graph \(G\) so that:

- \(G\) is 3-colorable iff \(F\) is satisfiable
Variable Part:
in 3-coloring, variable colors correspond to some truth assignment (same color as T or F).

Clause Part:
Add one 6 vertex gadget per clause connecting its ‘outer vertices’ to the literals in the clause.

Any truth assignment satisfying the formula can be extended to a 3-coloring of the graph.

Any 3-coloring of the graph colors each gadget triangle using each color.

Any 3-coloring of the graph has F opposite the O color in the triangle of each gadget.

Any 3-coloring of the graph has T at the other end of the blue edge connected to the F.
$3$-SAT $\leq^p 3$-Color

Any 3-coloring of the graph yields a satisfying assignment to the formula.